Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

2003-06

Packing in two and three dimensions

Martins, Gustavo H. A.

http://hdl.handle.net/10945/9859

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

PACKING IN TWO AND THREE DIMENSIONS

by

Gustavo H. A. Martins

June 2003

Dissertation Supervisor: Robert F. Dell

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2003	**3. REPORT TYPE AND DATES COVERED** Dissertation	
4. TITLE AND SUBTITLE: Title (Mix case letters) Packing in Two and Three Dimensions			**5. FUNDING NUMBERS**
6. AUTHOR(S) Martins, Gustavo H. A.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

This dissertation investigates Multidimensional Packing Problems (MD-PPs): the Pallet Loading Problem (PLP), the Multidimensional Knapsack Problem (MD-KP), and the Multidimensional Bin Packing Problem (MD-BPP). In these problems, there is a set of items, with rectangular dimensions, and a set of large containers, or bins, also with rectangular dimensions. Items cannot overlap (share the same region in space), and, when packed, must be completely located within the bin. We develop new theory for PLP, and apply it to the construction of new bounds, heuristics, and an exact algorithm. The bounds verify that the heuristics optimally solve 99.999% of PLP instances with up to 100 items; in the instances that the heuristics fail to solve optimally, their best solution differs from the optimum by only one item. Using our new PLP theory, we implement algorithms to solve orthogonal non-guillotine MD-KP instances and are the first to obtain exact solutions for some instances from the literature. Using these MD-KP algorithms, we develop the first exact algorithm for the orthogonal non-guillotine MD-BPP and are the first to obtain exact solutions to several instances from the literature.

14. SUBJECT TERMS Optimization, Combinatorics, Multidimensional Bin Packing Problem, Multidimensional Knapsack Problem, Pallet Loading Problem, Container Loading Problem, Cutting Stock.			**15. NUMBER OF PAGES** 176 **16. PRICE CODE**
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

PACKING IN TWO AND THREE DIMENSIONS

Gustavo H. A. Martins
Commander, Brazilian Navy
B.S., Escola Naval - Brazil, 1980
M.S., Naval Postgraduate School, 1993

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2003

Author: _____
Gustavo H. A. Martins

Approved by:

_____
Robert F. Dell
Associate Professor of Operations Research
Dissertation Supervisor

_____ _____
Gerald G. Brown, Distinguished Professor of Operations Research
Craig W. Rasmussen, Associate Professor of Mathematics

_____ _____
Alan R. Washburn, Professor of Operations Research
R. Kevin Wood, Professor of Operations Research

Approved by: _____
James N. Eagle, Chair, Department of Operations Research

Approved by: _____
Carson K. Eoyang, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This dissertation investigates Multidimensional Packing Problems (MD-PPs): the Pallet Loading Problem (PLP), the Multidimensional Knapsack Problem (MD-KP), and the Multidimensional Bin Packing Problem (MD-BPP). In these problems, there is a set of items, with rectangular dimensions, and a set of large containers, or bins, also with rectangular dimensions. Items cannot overlap (share the same region in space), and, when packed, must be completely located within the bin. We develop new theory for PLP, and apply it to the construction of new bounds, heuristics, and an exact algorithm. The bounds verify that the heuristics optimally solve 99.999% of PLP instances with up to 100 items; in the instances that the heuristics fail to solve optimally, their best solution differs from the optimum by only one item. Using our new PLP theory, we implement algorithms to solve orthogonal non-guillotine MD-KP instances and are the first to obtain exact solutions for some instances from the literature. Using these MD-KP algorithms, we develop the first exact algorithm for the orthogonal non-guillotine MD-BPP and are the first to obtain exact solutions to several instances from the literature.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This dissertation investigates Multidimensional Packing Problems (MD-PPs): the Pallet Loading Problem (PLP), the Multidimensional Knapsack Problem (MD-KP), and the Multidimensional Bin Packing Problem (MD-BPP). In these problems, there is a set of items, with rectangular dimensions, and a set of large containers, or bins, also with rectangular dimensions. Items cannot overlap (share the same region in space), and, when packed, must be completely located within the bin. In some cases, we want to maximize the value of items packed in a single bin (PLP and MD-KP). In others, we want to minimize the number of bins used to pack all items (MD-BPP).

MD-PPs arise when preparing a shipment of goods in rectangular boxes inside rectangular containers, or cutting smaller pieces from a larger stock (also known as the One-, Two-, or Three-Dimensional Cutting Stock Problem); allocating resources over time; planning the layout of newspaper pages; designing new computer chips; assigning processors in parallel computers; and many others. MD-PPs arise in the obvious military logistic applications and in others, e.g., campaign planning, where forces available and time are the two dimensions involved.

Most MD-PPs are known to be NP-complete and the related literature is rich in heuristics and approximation algorithms. There are efficient algorithms that yield optimal solutions for special cases of the general problem. The complexity of some of these special cases is still unknown.

In this dissertation, we implement new techniques, both exact and heuristic, to solve instances of PLP, MD-KP and MD-BPP. These techniques are based primarily on new theory developed for PLP.

Initially, we analyze the effect of restricting rotation and the effects of other common pattern restrictions on the number of bins required to solve instances of Two-Dimensional Bin Packing Problems (2D-BPP). We define the Minimum Size Instance (MSI) of an equivalence class of PLP, and show that every class has one and only one MSI. We also develop bounds on the dimensions of item and pallet in the MSI of a class, and show that a bound used for almost 15 years is incorrect. Applying the newly developed bounds on the MSI dimensions, we enumerate all 3,080,730 equivalence

classes with an area ratio (AR) bound no greater than 100 boxes. Previous work in this area considered only a subset of these classes.

We also present new bounding procedures for PLP. Some of these bounds rely on new relations among distinct equivalence classes of PLP (restricted and relaxed classes).

We develop new heuristics for PLP. The G5-heuristic computes optimal solutions to all instances of PLP with an AR bound of up to 51 boxes, and 99.999% of all possible instances with an AR bound of up to 100 boxes, differing from the optimum solution by at most one box in the 0.001% remaining instances. We develop three other heuristics to solve some instances not solved by the G5-heuristic or by heuristics from the literature.

After applying the new bounding procedures and heuristics, 6,952 equivalence classes of PLP with an AR bound of up to 100 boxes remain without a proven optimal solution. We develop an exact algorithm, the HVZ algorithm, to solve these remaining instances. With this new exact algorithm and the new bounds, we are able to identify the optimal solutions for the remaining instances.

We then extend the HVZ algorithm to develop the Diagonal Fill Algorithm (DFA) for the Two-Dimensional Knapsack Problem (2D-KP) and the Three-Dimensional Knapsack Problem (3D-KP). This is the first algorithm to solve instances of 2D-KP and 3D-KP from the literature, allowing 90º item rotation (orthogonal), and without the common guillotine cut restriction (non-guillotine).

We apply the algorithm for MD-KP within a branch-and-price algorithm to solve instances of MD-BPP. This is, also, the first algorithm to solve the orthogonal non-guillotine instances of MD-BPP from the literature. For cases where a good solution sooner is better than an optimal solution later, heuristic variations of our exact branch-and-price algorithm outperform existing heuristics on standard test instances.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank all the individuals and organizations that contributed to make my PhD program a success story. Even before mentioning a name, I already know that I won't be able to include everybody that should be acknowledged here, mainly because it would require doubling the number of pages in this document. I'm not sure if I deserve all this attention, but it really makes me proud.

I credit my parents for teaching me the importance of looking for the "why" behind the "what" in all things.

Gláucia and Letícia, my wife and daughter, thank you for all the support and understanding, especially when I was completely absent, although being physically at home.

I'm only reaching the present education level because of the Brazilian Navy's confidence in me. I'll do my best to show that investing in postgraduate education is a strategy that should be continued.

I also express my gratitude to:

- The faculty of the Operations Research and Mathematics Departments of the NPS, for their fundamental contribution to my education.
- Professors Alan Washburn, Jerry Brown, Kevin Wood and Craig Rasmussen, not only for everything they taught me in lectures, but also for their orientation and helpful comments, in making this document look like a dissertation. Special thanks to Professor Washburn, who helped clarify Theorems IV.1 and IV.2.
- Reinaldo Morabito and Guntram Scheithauer, for contributing their expertise in packing problems.
- My friends and colleges at the "Centro de Análises de Sistemas Navais", for their encouragement and continued interest in my research.

- Cindy Graham and all others in the International Programs Office, who always strive to improve the quality of life for NPS International Students.
- My fellow PhD students, Mehmet Ayik, Izumi Kobayashi, and Tom Cioppa, now with their programs completed, for their encouragement when things looked difficult. Special thanks to Kristi and Scott Frickenstien, my sponsor in Monterey, for welcoming me as part of their family.
- Ron Del Presto and all the others in the Glasgow Computer Support Group for keeping me "logged in."

I express a special note of gratitude to Professor Rob Dell, my dissertation supervisor, who always went the extra mile to help me complete this demanding task. Professor Dell not only provided the motivation for this research, pointed the directions to follow, and corrected any deviations, but he also took over the dissertation reviews after I returned home with only an incomplete draft in hand.

I. MULTIDIMENSIONAL PACKING PROBLEMS

A. INTRODUCTION TO MULTIDIMENSIONAL PACKING PROBLEMS

This dissertation investigates Multidimensional Packing Problems (MD-PPs): the Pallet Loading Problem (PLP), the Multidimensional Knapsack Problem (MD-KP), and the Multidimensional Bin Packing Problem (MD-BPP). In these problems, there are a set of *items*, with rectangular dimensions, and a set of large containers, or *bins*, also with rectangular dimensions. Items cannot overlap (share the same region in space), and, when packed, must be completely located within the bin. In some cases, we want to maximize the value of items packed in a single bin (PLP and MD-KP). In others, we want to minimize the number of bins used to pack all items (MD-BPP).

MD-PPs are encountered when preparing a shipment of goods in rectangular boxes inside rectangular containers, or when cutting smaller pieces off some larger stock (also known as the *One-*, *Two-*, or *Three-Dimensional Cutting Stock Problem*); allocating resources over time; planning the layout of newspaper pages; designing new computer chips; assigning processors in parallel computers; and other applications. MD-PPs arise in the military in obvious logistic applications and in others, e.g., campaign planning, where forces available and time are the two dimensions involved. There are many other applications too (e.g., Dyckhoff [1990]).

Most MD-PPs are known to be NP-complete [Garey and Johnson 1979], and the related literature is rich in heuristics and approximation algorithms. Efficient algorithms that yield optimal solutions, for some restrictions, or special cases, of the general problem also exist. The complexity of some of these special cases is still unknown.

In this dissertation, we implement new techniques, both exact and heuristic, to solve instances of PLP, MD-KP and MD-BPP. These techniques are based primarily on new theory developed for PLP.

B. PROBLEM DEFINITION

The basic *Cutting and Packing Problem* (C&PP) involves two sets, one set of small objects, *items*, and one set of large objects, *bins*. All items and bins have fixed shapes. The problem is how to arrange items within the bins in order to optimize some function, subject to constraints restricting items not to overlap, and to ensure the items are fully contained inside the bin. There is no difference, in terms of solution strategies, between packing and cutting problems (e.g., Dyckhoff [1990]).

Additional restrictions on shapes and quantities of items and bins available, restrictions on certain arrangements, the selection of objective functions, the availability of complete information, and the number of dimensions involved generate the variations on C&PP encountered in the literature. MD-PPs are special cases of C&PP where both items and bins have rectangular shape. We use 2D and 3D when referring specifically to two- and three-dimensional problems.

Some of the most studied MD-PPs are Multidimensional Bin Packing, Multidimensional Knapsack, Multidimensional Strip Packing, and Pallet Loading.

- *Multidimensional Bin Packing Problem* (*MD-BPP*). A 2D-BPP instance consists of a list I of rectangular items with dimensions (l_i, w_i) and demand d_i for all $i \in I$ as well as a list B of bins with dimensions (X_b, Y_b) and cost C_b for all $b \in B$. For a 3D-BPP instance, the items have dimensions (l_i, w_i, h_i) and the bins have dimensions (X_b, Y_b, Z_b). The objective is to minimize the total cost of the bins used to pack all items. If only one type of bin is available, with dimensions (X, Y) or (X, Y, Z), the objective becomes the minimization of the number of bins used. 3D-BPP is also called the *Multi-Container Loading Problem*.
- *Multidimensional Knapsack Problem* (*MD-KP*). In an instance of MD-KP, items have values v_i, $i \in I$, and only one bin is available. The objective is to pack, in the bin, the subset of items with maximum value. If the value of an item is proportional to its area, or volume, then an equivalent objective is to minimize the unused, or wasted, region in the bin. If $d_i = \infty, i \in I$, then the

problem is *Unconstrained*; otherwise, it is *Constrained*. This problem is different from the *Multi-Constraint Knapsack Problem*, also called the *Multi-knapsack Problem*, where the constraints are defined over possibly independent dimensions. Lin [1998] presents a recent survey on multi-constraint knapsack problems. 3D-KP is also called the *Container Loading Problem*.

- *Multidimensional Strip Packing Problem* (*MD-SPP*). In MD-SPP, one of the dimensions of the bin is unconstrained (the length in 2D or height in 3D) and we want to minimize the unconstrained dimension required to pack all items.
- *Pallet Loading Problem* (*PLP*). PLP is encountered when trying to maximize the number of identical boxes (items) with dimensions $a \times b$, loaded in a pallet (bin) with dimensions $X \times Y$ where each item has a "this side up" type of restriction (e.g., Bischoff and Dowsland [1982]). In this case, items are loaded in vertical layers, of the same height, and the problem reduces to finding the two-dimensional arrangement which maximizes the number of packed items in a layer. PLP is an example of unconstrained 2D-KP with only one type of item to pack.

Although MD-KP and MD-SPP are distinct, an algorithm for MD-KP in n dimensions can be used to solve MD-SPP in n dimensions. If an algorithm for 2D-KP is available, we can fix the width of the bin and try different values for the length. The solution with minimum length, such that all items are packed, is the solution to 2D-SPP. Also, algorithms for MD-KP and MD-SPP can determine whether all items on a list of items can be packed in a single bin. When using an algorithm for MD-KP, we use the area, or volume, of the items as their values. With the algorithm for MD-SPP, we solve for the minimum dimension, and if the solution does not exceed the corresponding dimension on the bin, then the packing is feasible.

When packing bins, we require all items to be pushed as much as possible to the left, front, and bottom of the bin. An item cannot be pushed any further if it touches the border of the bin or another item. This packing pattern is called a *Normal Packing Pattern* [Christofides and Whitlock 1977].

C. TYPOLOGY OF CUTTING AND PACKING PROBLEMS

In an approach similar to the characterization of queuing problems, Dyckhoff [1990] proposes "a consistent and systematic approach for a comprehensive typology integrating the various kinds of [packing and cutting] problems." Dyckhoff identifies nine main characteristics of this family of problems: *Dimensionality*, *Quantity Measurement*, *Shape of Figures*, *Assortment*, *Availability*, *Pattern Restrictions, Assignment Restrictions*, *Objectives*, and *Status of Information and Variability*.

Dimensionality: What is the minimum number of dimensions necessary to describe the geometry of the packing patterns? This is not necessarily the number of dimensions of the items and bins involved. In some cases, although dealing with three-dimensional items, we are only interested in the arrangement of these items on a plane, reducing the problem to two-dimensional. An example is PLP, where the same two-dimensional pattern is repeated on each vertical layer of the pallet. This dissertation covers two- and three-dimensional problems.

Quantity measurement: What is the objective function? It can be the number of bins used to pack the items (as in MD-BPP) or it can be the length or height of a continuous bin (as in MD-SPP). The *Earliest Completion Time Problem*, where a set of tasks is assigned to multiple workers to minimize the total time to complete the tasks, is an example of a continuous quantity, time, being optimized. In instances of MD-PP studied in this dissertation, we minimize the number of bins used (MD-BPP), or we maximize the value of packing one bin (MD-KP and PLP).

Shape of figures: What shapes do items and bins have? They can be non-rectangular, and even non-regular. The *Marker Layout Problem*, encountered in the garment industry, is an example where items have irregular shapes (e.g., Dowsland and Dowsland [1995]). In this problem a large piece of cloth is cut in smaller irregular shaped parts so as to minimize leftovers.

The orientation of items inside bins is also addressed under this characteristic. This dissertation only covers rectangles, in two dimensions, and orthogonal parallelepipeds, in three dimensions. Also, items can be rotated, but their sides must be parallel to the sides of the bin.

Assortment: What is the number of different shapes available for items and bins? In the most common formulation of MD-BPP, all bins have the same shape. In PLP, all items have the same shape, but may differ in their orientation. In "The Dissection of Rectangles into Squares" [Brooks et al 1940], items are squares, but each with a different size. This dissertation analyzes problems with items and bins having a small assortment of different shapes.

Availability: Are there constraints on the available quantities for items and bins? MD-KP is an example where the available number of bins is limited. The literature has references on constrained and unconstrained problems, conditioned on the existence of bounds on the quantity of items to be packed (e.g., Beasley [1985a]). This dissertation addresses problems with various restrictions on availability of items and bins.

Pattern restrictions: What patterns or geometric combinations can the items take inside the bins? As an example, it is common to assume in the cutting stock problem that all cuts are guillotine cuts. A *guillotine cut* divides a piece of larger stock in two smaller pieces with a cut from one side to the other. The recursive application of guillotine cuts forms a *guillotine cut pattern*. Figure I.1 provides an example of a two-dimensional pattern produced with guillotine cuts. This dissertation analyzes the effect of some pattern restrictions.

Assignment restrictions: Are there restrictions on the possible assignments of the items? Compatibility of items being packed together may limit the number of feasible assignments. In other cases, information on all items is not available when deciding how to pack an item. This happens, for example, when loading the items as they are being delivered: These are "on-line" problems. This dissertation considers only "off-line" problems, in which shapes and quantities of all items and bins are known when selecting the packing pattern.

Objectives: Is the objective function to be maximized or minimized? There might be a cost associated with the number, or type, of bins used (as in MD-BPP). The total quantity of "scrap," or unused material, after cutting items is another common criterion to be minimized (as in MD-KP). As stated above, this dissertation addresses both the

minimization of the number of bins used, and the maximization of the value of items packed in one bin.

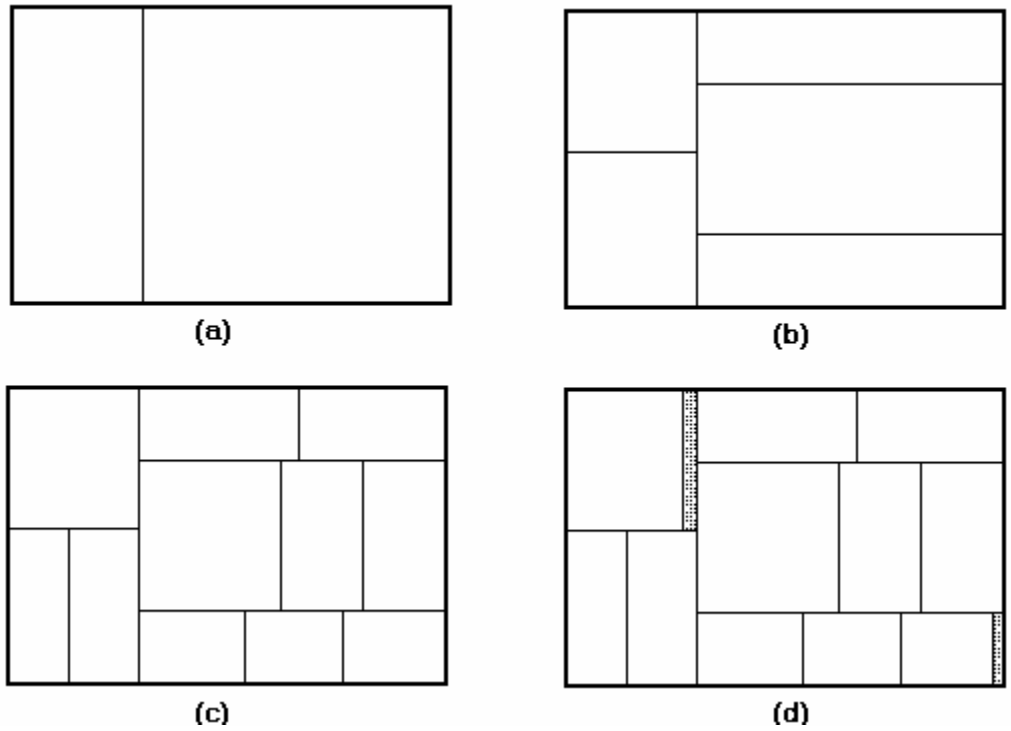

Figure I.1 *An example of a pattern with guillotine cuts.*
At each step of the cutting process, a larger rectangle is slit in two smaller rectangles, with a cut from one side to the other. (a) The original stock is cut in two smaller pieces; (b) and (c) subsequent cuts are performed; and (d) pieces are trimmed to their correct size.

Status of information and variability: Are the exact shapes and quantities known with certainty? If only the distribution of the items is known, one of the most common objectives is to maximize the expected number of items packed. Coffman and Shor [1990] and Coffman and Lueker [1991] exemplify work in this area. This dissertation investigates only the deterministic version of the problem.

After considering these nine characteristics, Dyckhoff proposes 96 problem types identified by the four characteristics of dimensionality, kind of assignment, assortment of bins, and assortment of small items.

Dimensionality:

(1) One-dimensional.

(2) Two-dimensional.

(3) Three-dimensional.

(N) N-dimensional, N > 3.

Kind of Assignment:

(B) All bins and a selection of items.

(V) A selection of bins and all items.

Assortment of bins:

(O) One bin.

(I) All bins are identical.

(D) Bins have different sizes.

Assortment of Small Items:

(F) Few items, with different shapes and sizes.

(M) Many items of many different shapes and sizes.

(R) Many items of relatively few different (non-congruent) shapes and sizes.

(C) Items with congruent sizes.

Using this typology, the problems in this dissertation, MD-BPP, MD-KP, and PLP, are identified as:

- MD-BPP: $2/V/\cdot/\cdot$ or $3/\mathrm{V}/\cdot/\cdot$. For example, a 3D-BPP with identical bins, and a two different items types fall in $3/V/I/F$.
- MD-KP: $2/B/O/\cdot$ or $3/B/O/\cdot$, where we must pack a selection of items in one bin.
- PLP: $2/B/O/C$, a special case of unconstrained 2D-KP.

Some of the characteristics of problems approached in this dissertation, namely the restriction of rectangular shapes for both items and bins, and finite dimensions for bins, are not distinguished within this typology.

D. LITERATURE REVIEW FOR MD-PP

The basic C&PP is known to NP-hard, with some specific cases shown to be NP-complete in the strong sense. This encourages the analyst to concentrate his work in ways to identify special situations where the solution might be obtained in less time than would be required by the total enumeration of all possible solutions, or to develop heuristics or approximation algorithms [Garey and Johnson 1979].

The operations research (OR) literature is rich in references to C&PP under titles like Trim Problem, Loading Problem, Floor Layout Problem, Job Scheduling, and Resource Allocation. Sweeney and Paternoster [1992] provide a comprehensive bibliography on the subject up to 1990, with more than 400 references. Lodi et al [2002a] survey two-dimensional problems. Bischoff and Wäscher [1990], Dyckhoff and Wäscher [1995] and Wang and Wäscher [2002] edit special issues of the *European Journal of Operations Research* on the subject. The online library of the Special Interest Group in Cutting and Packing [SICUP 2002] is a source of information on recent C&PP work.

The first references in the literature to this family of problems are related to the problem of decomposing a square into incongruent sub-squares; many mathematicians in the past considered this to be an unsolvable problem. Brooks et al [1940] are the first to model the problem as a network. Meschkowski [1966] shows that decomposing rectangles into squares can also be modeled as a network flow problem. Gambini [1999] exemplifies present work in this square decomposition problem. See also Biró and Boros [1984] and Lins et al [2002] for network-based models for MD-PP.

In 1939, Kantorovich (Kantorovich [1960]) analyzes a problem he names *Minimization of Scrap*. This is the one-dimensional cutting stock version of MD-BPP, under a linear programming framework. In his work, he assumes that a list of feasible cutting patterns is available beforehand, and approaches the decision of which patterns to select.

Gilmore and Gomory [1961] present a method for generating "on the fly" the best possible patterns for the one-dimensional cutting stock version of MD-BPP, using column generation. This is a heuristic procedure, because it uses a rounding scheme for leftover fractional solutions. However, in large instances of the problem, the loss in optimality due to rounding is expected to be small. Gilmore and Gomory [1965] extend their own work to higher dimensions, but only consider patterns generated by *multistage guillotine cuts*. At each stage of the multistage cutting process, guillotine cuts are performed in only one direction. Beasley [1985a] and Farley [1990], among others, also investigate algorithms based on multistage guillotine cuts.

Barnet and Kynch [1967] analyze the packing of $a \times 1$ rectangles within a larger rectangle, and present possible solutions to this problem. De Bruijn [1969] studies the packing of n-dimensional bricks within a large n-dimensional box. Brualdi and Foregger [1974] extend the work on *harmonic* bricks. An $a_1 \times a_2 \times \ldots \times a_n$ brick is said to be *harmonic* if the integers $a_1, a_2, \ldots, a_n$ can be rearranged into $a'_1, a'_2, \ldots, a'_n$, such that $a'_1 | a'_2 | a'_3 | \ldots | a'_n$ (where $c | d$ signifies that c divides d). Barnes [1979] uses this work to propose a new bound for PLP, and to show that for "sufficiently large" problems, this bound is exact. A problem is considered *sufficiently large* if the dimensions of the pallet are significantly larger than the dimensions of the item (i.e., $\min\{X, Y\} \geq a * b$).

Christofides and Whitlock [1977] propose a tree-search algorithm for 2D-KP with guillotine cuts and with items restricted to a fixed orientation. In their formulation, guillotine cuts can be performed in both directions; their algorithm is based on generating all feasible cutting patterns. They also propose the use of Normal Patterns, representing left-bottom justified packing patterns. Herz [1972] presents the same idea with the name of *Canonical Dissections*.

Wang [1983] presents a different approach for enumerating the feasible guillotine patterns in 2D-KP. Her algorithm generates patterns by "successively adding the rectangles to each other." See also Oliveira and Ferreira [1990] and Daza et al [1995].

Beasley [1985b] proposes the first exact algorithm for solving 2D-KP without the guillotine cut restriction (non-guillotine 2D-KP). His approach is based on a branch-and-bound tree search using lagrangean relaxation of a *binary integer program* (BIP) to generate bounds. In his formulation the orientation of items is fixed, although he proposes an approach to deal with the situation of 90º rotations. Christofides and Hadjiconstantinou [1995] use a similar idea, with an improved formulation, but also with fixed orientation.

Baker et al [1980] propose an approximation algorithm for 2D-PP, and compute asymptotic performance bounds for this algorithm. Coffman et al [1980], Baker and Schwarz [1983], and Kenyon and Rémila [2000] also consider approximation algorithms for 2D-PP. Chung et al [1982] use a hybrid procedure, based on 2D-PP, in an approximation algorithm for 2D-BPP with fixed orientation, and show that the solutions

obtained are, in the worst case, larger than the optimal solution by a factor slightly larger than 2.

Several authors investigate different types of heuristics for C&PP. George and Robinson [1980] propose a heuristic to pack three-dimensional boxes in containers, and report using the heuristic to pack "about 800 boxes of up to 20 different types in each shipment." Gehring et al [1990], and Bischoff et al [1995] also describe heuristics for 3D-BPP. Bischoff and Marriott [1990] investigate the effect of certain parameters, such us sorting and ranking policies, on the performance of two existing heuristics, and conclude by proposing a new combined heuristic. Kröger [1995] and Gómez and de la Fuente [2000] propose heuristics with genetic approaches. Healy and Moll [1996] describe a heuristic based on local search. Faina [2000] proposes an algorithm based on simulated annealing. Hopper and Turton [2001] investigate hybrid heuristics.

Li and Cheng [1990] propose approximation algorithms for 3D-SPP, which guarantee results not worse than 3.25 times the optimum height of packed items. Their algorithms divide items into subgroups based on size, and apply different packing procedures to each subgroup. Miyazawa and Wakabayashi [1997] and [2000] improve those algorithms. Li and Cheng [1992] study the on-line version of 3D-SPP.

Martello et al [2000] analyze new bounds for 3D-BPP, and present both exact and approximation algorithms. Their exact algorithm is based on a two-level decomposition idea, also used for 2D-BPP [Martello and Vigo 1998], and can be traced back to the original ideas of Gilmore and Gomory [1961]. Chen et al [1995] propose a different approach, a *mixed integer program* (MIP) for 3D-BPP, although only a small instance is solved.

Heuristics for solving 3D-BPP are proposed by Ivancic et al (as reported by Bischoff and Ratcliff [1995]), Bischoff and Ratcliff [1995], and Terno et al [2000], among others. Terno et al [2000] uses the 4-block heuristic as a subroutine in a new heuristic, the *Parallel Generalized Layer-Wise Loading Approach* (PGL-approach), to solve the *Multi-Pallet Loading Problem*, a special case of 3D-BPP. Current work in the field is represented in Eley [2002], Lins et al [2002b], and Pisinger [2002].

E. COMMON VARIATIONS OF MD-BPP

We analyze some of the most common variations of the general MD-BPP, with a discussion of the implications of some simplifications and restrictions. These variations are usually related to the orientation of items (if items can be rotated and how), the types of patterns accepted when packing or cutting the items, and restrictions on the number of items being packed.

We mainly bound the effects of these variations on large instances. Therefore, let $A(L)$ be the number of bins used to pack a list L of items, when applying algorithm A. Let $OPT(L)$ be the corresponding minimum number of identical bins necessary to pack the same list L. Define $R_A(L) \equiv A(L)/OPT(L)$, $R_A^n \equiv \max\{R_A(L) \mid OPT(L) = n\}$, and $R_A^\infty \equiv \lim_{n\to\infty} \sup R_A^n$. R_A^∞ measures the *asymptotic worst case behavior* of A, also known as the *asymptotic performance bound* for A.

1. Orientation

When positioning items inside a bin, it might be useful to rotate some items in order to obtain a better solution. If two items, with the same shape and size, but with different orientations, are indistinguishable in the context of the problem in question, we say that any orientation is allowed. If a different orientation causes an item to be treated as a different item, we say that the problem is *oriented* [Baker et al 1980]. It is also possible that the edges of the items must be loaded parallel to the bin's edges, and only 90º rotations are allowed. This restriction is usually dictated by packing or cutting considerations, and is called the *Orthogonal Packing Problem* [Baker et al 1980].

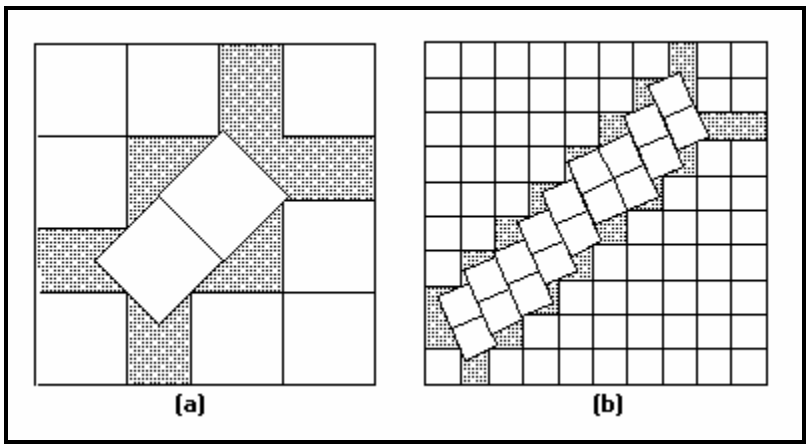

Figure I.2 *Examples of unit squares packed in a larger square.*
All small squares have a unit side, and the larger square bins have the minimum known dimension to pack the given number of unit squares (after Friedman [2000]).

Erdös and Graham [1975] show that even when packing unit squares in a larger square, it is possible to improve the solution by allowing rotation. Figure I.2 (after Friedman [2000]) illustrates this situation. In Figure I.2 (a), 10 squares are packed in a square with side $3+1/\sqrt{2}$, and Figure I.2 (b) has 86 squares packed in a square with side $(17+\sqrt{7})/2$. Without rotation, only 9, and 81, respectively, unit squares can be packed in the larger squares. A packing of 10, or 86, unit squares in square bins with sides smaller than $3+1/\sqrt{2}$, and $(17+\sqrt{7})/2$, respectively, is unknown [Friedman 2000].

In most formulations, even 90º rotations are disallowed. Although adopted in many packing and cutting problems, it is often only realistic when cutting pieces of wood, or glass, where the orientation of the pieces are important to the final product. In such cases, pieces with the same dimensions, but different orientations, are considered as different pieces.

Christofides and Whitlock [1977], besides requiring guillotine cuts in their tree search algorithm, also consider the orientation of the pieces to be fixed. Baker et al [1980] and Chung et al [1982] assume that the orientation of the smaller rectangles cannot be changed in their approximation algorithms, and the asymptotic worst case bounds computed are valid only under that assumption. Even more recent work, e.g., Christofides and Hadjiconstantinou [1995], Martello and Vigo [1998], Martello et al [2000], still require fixed orientation.

Except for the obvious conclusions drawn from the work of Erdös and Graham [1975], we do not find any literature that analyzes the effect of restricting item orientation on MD-BPP, when compared with solutions obtained without this restriction.

Theorem I.1: Let A be an algorithm for 2D-BPP that requires all items to have fixed orientation, while OPT accepts 90º rotations. Then $R_A^\infty \geq 2$.

Proof: Consider the problem of packing n rectangles with dimensions 3 × 2 inside bins with dimensions 5 × 3. With fixed orientation, n bins are necessary. But when allowing 90º rotations, only $n/2$ bins are necessary. Therefore, $R_A^n \geq n/(n/2) = 2$ and $\lim_{n\to\infty} R_A^n \geq 2$. **Q.E.D.**

As shown above, algorithms considering only fixed orientation might obtain solutions that are twice as large as those obtained by orthogonal algorithms. But allowing only 90º rotations may also substantially increase the number of required bins, as we see below.

Theorem I.2: Let B be an orthogonal algorithm for 2D-BPP, i.e., an algorithm that provides only for 90º rotations, while OPT accepts any rotation. Then $R_B^\infty \geq 2$.

Proof: Consider the problem of packing $n = k^2$, k integer, rectangles with length $1+\varepsilon$ and width ε, with $\varepsilon = 1/k$, in bins with corresponding dimensions 2 × 1. In this case, because of the length of the items, we cannot rotate them 90º within the bin, and only k items are packed in each bin, with k bins being necessary. But if we rotate the items as close as possible to the vertical, as in Figure I.3, for sufficiently small ε, $2k-2$ items are packed, and $k/2+1$ bins are necessary ($k/2$ bins with $2k-2$ items, and one bin with k items). Then $R_B^\infty \geq \lim_{k\to\infty} k/(k/2+1) = 2$. **Q.E.D.**

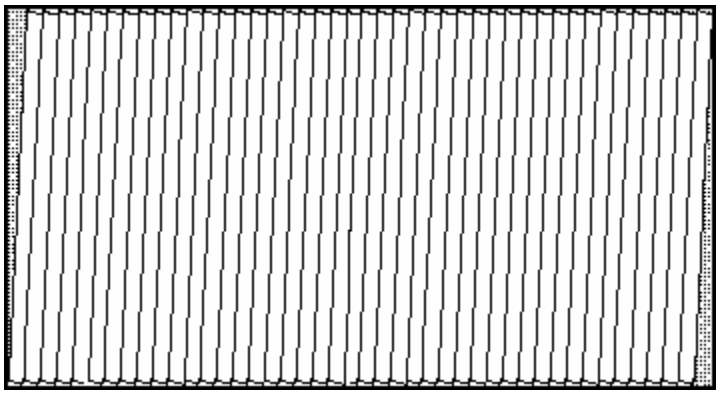

Figure I.3 *Packing items rotated by an angle close to* 90^{o}.
This Figure shows $2k - 2$ *items, with dimensions (*$1 + \varepsilon$*)* $\times$ ε*, and* $\varepsilon = 1/k$*, rotated by an angle close to* 90^{o}*, packed in a bin with dimensions* 2×1.

In both cases above, if a third dimension is added, with value 1, for both items and bins, we observe the results above are also true for 3D-BPP algorithms.

2. Pattern Restrictions

The most common type of pattern restriction is guillotine cuts (Figure I.1). Initially addressed by Gilmore and Gomory [1965], guillotine cuts have been further studied by several authors (e.g., Christofides and Whitlock [1977], Wang [1983], Oliveira and Ferreira [1990]).

If we consider the problem of packing 4×3 items inside 7×7 bins, with 90º rotations without a guillotine cut restriction, 4 items can be packed in each bin, as shown in Figure I.4. With guillotine cut restrictions, only 3 items can fit in each bin, increasing by 1/3 the number of bins required. Therefore, if C is an algorithm for 2D-BPP with guillotine cut restriction, then $R_C^{\infty} \geq 4/3$.

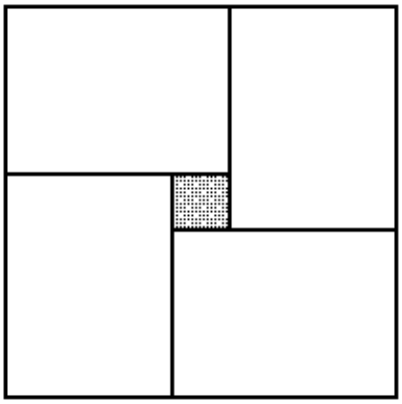

Figure I.4 *Optimal packing with non-guillotine cuts allowed of* 4×3 *items in a* 7×7 *bin. This layout is not a guillotine cut pattern, and there is no guillotine cut pattern that packs 4 items.*

Another type of pattern restriction is the 1st-Order Non-Guillotine Cutting Pattern. Arenales and Morabito [1995] define *1st-Order Non-Guillotine Cutting Pattern*, or *1st-Order Pattern*, as a pattern generated only by successive guillotine and/or 1st-order cuts. A cut is a *1stOrder Non-Guillotine Cut*, or *1st-order*, if it produces five new rectangles arranged in such a way as not to form a guillotine cutting pattern, as shown in Figure I.5 (a).

Figure I.5 (b) shows an example of a 1st-order pattern, while Figure I.5 (c) shows an example of a pattern that is not a 1st-order pattern.

We find 1st-order patterns provide optimal solutions to all PLP instances with less than 52 items (shown in Chapter IV). Therefore, we explore some instances using items with three different sizes, and a total of seven items packed in each bin.

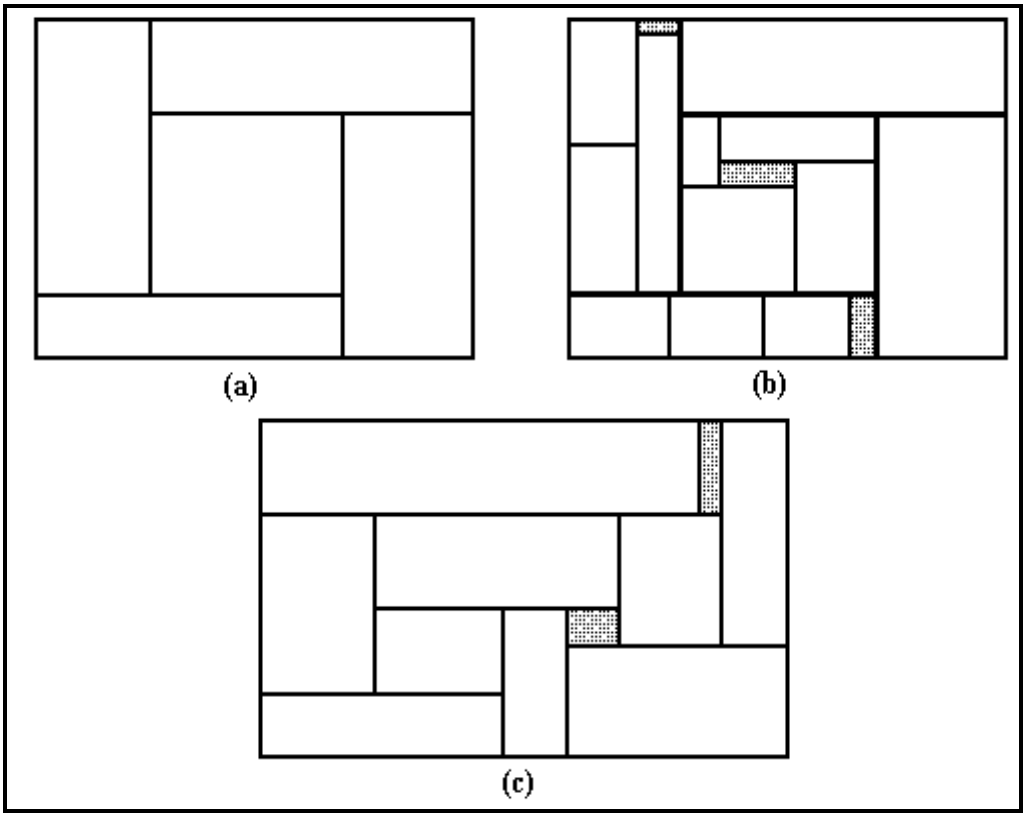

Figure I.5 *Examples of 1st-order packing patterns.*
(a) represents a 1st-order cut, resulting in five rectangles; (b) is a 1st-order cutting pattern, because it can be obtained by recursively performing 1st-order and guillotine cuts; and (c) is an example of a cutting pattern of higher order.

Theorem I.3: If D is an algorithm based on 1st-order patterns, then $R_D^{\infty} \geq 27/25$.

Proof: Consider the problem of packing 3*n* 29 × 11 items (*I*), 2*n* 28 × 23 items (*II*), and 2*n* 41 × 5 items (*III*), in bins with dimensions 69 × 39. Using the non 1st-order pattern shown in Figure I.6, it is possible to pack 7*n* items in *n* bins.

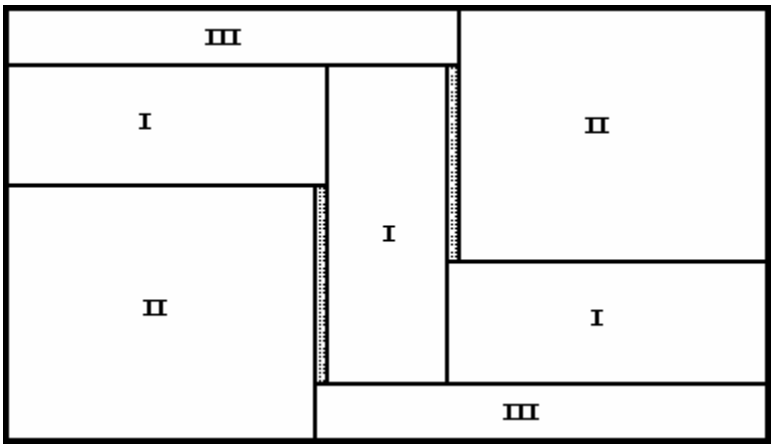

Figure I.6 *Non 1st-order pattern used in the proof of Theorem I.3.*
This is a non 1st-order packing pattern, with three 29 × 11 items (I), two 28 × 23 items (II), and two 41 × 5 items (III), in a bin with dimensions 69 × 39.

If only 1st-order patterns can be used, then this optimal pattern is infeasible and we must investigate other feasible patterns. After verifying that a pattern is feasible, the decision on how many bins to fill with a given pattern depends only on how many instances of each item are present in the pattern. If two patterns have the same number of instances of each item, then they are considered equivalent, even if the arrangement is distinct. We only consider patterns that are maximal, in the sense that no additional items can be packed in the bin. Table I.1 presents the maximal feasible 1st-order patterns in the problem. The columns correspond to different patterns; the rows to items; and the entries are the number of copies of each item present in each pattern.

	Patterns													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
I	7	6	5	5	4	3	3	3	2	2	2	1	1	1
II	0	0	1	0	1	2	1	0	2	1	0	3	2	0
II	1	2	1	3	3	1	3	5	2	5	7	0	4	7

Table I.1 *Packing patterns used in the proof of Theorem I.3.*
Columns correspond to different 1st-order patterns, the rows correspond to different items, and the entries are the number of copies of each item present in each pattern. The column numbered 1 corresponds to a pattern with seven items of type I, and one item of type III.

A linear relaxation of the problem of selecting the best set of 1st-order patterns to minimize the number of bins requires $4n/25$ bins with pattern 5, $18n/25$ bins with pattern 6, and $5n/25$ bins with pattern 13. Because a linear relaxation provides a lower bound on

the solution for the integer program, at least $27n/25$ bins are required, and $R_D^\infty \geq 27/25$. **Q.E.D.**

The last type of pattern restriction considered in this dissertation is the requirement of connectivity of items of the same type. In 2D-BPP, when *connectivity* is required, all instances of the same item in a bin must be packed together, i.e., they must share a side. George and Robinson [1980] "choose a box type and fill as many complete columns" as possible. Terno et al [2000] also require connectivity.

Theorem I.4: If E is an algorithm requiring connectivity, then $R_E^\infty \geq 4/3$.

Proof: In Figure I.7 (a) we observe the optimal pattern used when packing $2n$ 3×3 items and $2n$ 4×2 items in 7×5 bins, if connectivity is not required. In this case, n bins are necessary. But if connectivity is required, an optimal solution uses n bins packed with the pattern in Figure I.7 (b), and $n/3$ bins with a pattern like Figure I.7 (c). Therefore, $R_E^\infty \geq 4/3$. **Q.E.D.**

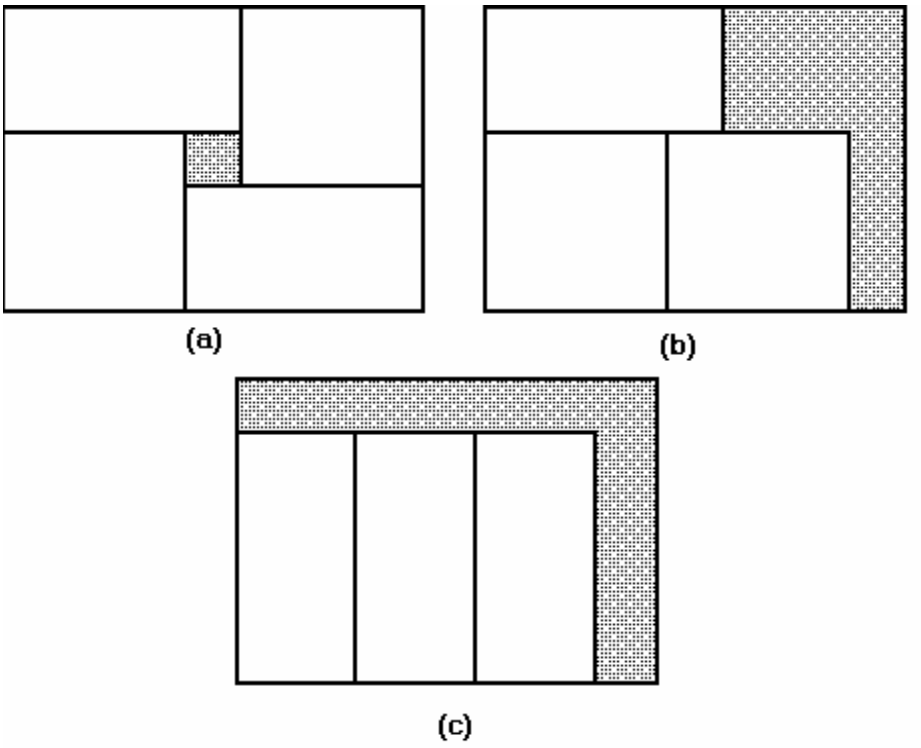

Figure I.7 *Example of packing layouts of 2n 3×3 items and 2n 4×2 items in 7× 5 bins. In (a) connectivity is not required, but in (b) and (c) connectivity is required.*

F. COMPLEXITY OF MD-PP

Garey and Johnson [1979, pp. 124-127] show that the one-dimensional bin packing problem (1D-BPP) is NP-complete in the strong sense. Because the one-dimensional problem can be transformed into higher dimensional cases by adding extra, fixed dimensions, MD-BPP is also strongly NP-complete. The above transformation does not apply to some of the special cases of MD-BPP, and results concerning the complexity of these cases rely on other approaches. As an example, if the items or bins are square, then the transformation is not straightforward, because fixing one of the sizes would also fix the other, turning it into a trivial problem, at least for orthogonal MD-BPP.

Li and Cheng [1989] show that 2D-KP remains strongly NP-complete when packing squares in a rectangular bin, or packing rectangles into a square bin. Leung et al [1990] prove that packing squares inside a square bin also remains strongly NP-complete. But their proof, in which the *3-Partition Problem* [e.g., Garey and Johnson 1979] is transformed into a 2D-KP, requires a large number of different size square items. Because the orthogonal problem with only one square size is trivial, and the general cases are strongly NP-complete, there must be a number of distinct sizes, greater than 1, where the problem becomes hard.

The transformation of 1D-BPP into MD-BPP helps to prove complexity results for the latter, but it does not directly offer insight into the real difficulties of solving multidimensional problems. In the 1D-BPP, the solution is given by partitioning the set of items into subsets, with each subset assigned to a bin. The only requirement is that the sum of the length of the items in each subset does not exceed a given bound, the capacity of the bin. Because addition is commutative, the order in which the items are packed is irrelevant. This is easily observed when considering the one-dimensional version of PLP. If the bin has length X and the item length l, then the number of items that can be packed is $\lfloor X/l \rfloor$.

In the multi-dimensional case, even when the subsets are given, it is still necessary to verify the feasibility of packing the items. To verify feasibility, we must solve a MD-KP, with the value of each item given by its area. This MD-KP, alone, is NP-complete.

The solution of MD-BPP must include a partition of the items into subsets assigned to each bin, and the *arrangement*, or packing pattern, of these items inside each bin. The most common way to represent this arrangement is to assign each item to a position relative to some reference point in the bin. For example, each item can be assigned a pair of coordinates (x_i, y_i), corresponding to the horizontal and vertical distance to the reference point in the bin. If only normal patterns are considered, then the number of positions an item can take in the bin is finite, although usually large [Christofides and Whitlock 1977].

Another approach to represent the arrangement is using *Sequence Pairs* [Murata et al 1995], where two ordered lists represent the relative position between pairs of items. Each of these two lists corresponds to permutations of $\{1,...,n\}$, where n is the number of packed items. There are four possible relative positions between items i and j : j is to the right of i, i is to the right of j, j is above i, or i is above j.

Graphs can also represent packing patterns (e.g., Biró and Boros [1984] and Lins et al [2002]). The main problem with this approach is that different representations can lead to the same arrangement, as given by the coordinates of each item [Christofides and Hadjiconstantinou 1995]. This adds to the complexity of search algorithms, even when the number of solutions is small.

If we consider only orthogonal problems with normal packing patterns, the decisions to solve MD-BPP can be divided in four main types:

***Partition*:** Which item goes in which bin? The number of possible assignments increases as $2^{|I|}$. This decision is always present in MD-BPP, and the complexity can only be reduced by adding constraints on combinations of items that can be packed together in a bin.

***Order*:** After choosing a partition, an algorithm must select the order in which the items are packed, and the number of options goes up with the factorial of the number of items in the partition. In PLP, because there is only one type of item, there is no ordering decision.

***Orientation*:** Each rectangular item can have two distinct orientations in two dimensions, or up to six orientations in three dimensions. Because each item can usually be rotated independently of the other, there are an exponential number of possibilities.

***Relative Position*:** After deciding the ordering and the orientation, it is still necessary to select the position of each item being packed relative to all items previously packed in the same bin. The number of possible positions for placing an item may increase with the number of patterns that can be used with the items already packed, and the number of different patterns increases with the factorial of the number of items in the partition. The complexity of this positioning is simplified by rules like "left-most downward" (e.g., Christofides and Hadjiconstantinou [1995]).

Most heuristics and approximate algorithms in the literature reduce the size of the solution space by using a sorting rule to decide the ordering of items, fixing the orientation of the items, or packing items according to layers (e.g., Coffman et al [1980], Chung et al [1982], Li and Cheng [1990]).

Although exact algorithms for most variations of MD-BPP have exponential run time, they can significantly differ in terms of complexity because of the way the solutions are represented. For example, the algorithm for oriented 2D-SPP in Murata et al [1995], using sequence pairs, has complexity $O(n^8 n!^2)$ [Xu et al 1997], while the MIP analyzed by Chen et al [1995], when applied to oriented 2D-SPP, has complexity $O(2^{2n})$, where n is the number of items to pack.

G. OUTLINE OF THE DISSERTATION

This chapter introduces MD-BPP, MD-KP, PLP, and related problems.

Chapter II reviews previous work on PLP, including a review on the state of the art of algorithms and bounding procedures. In Chapter III, we define the Minimum Size Instance (MSI) of an equivalence class of PLP, and show that every class has one and only one MSI. We also develop bounds on the dimensions of box and pallet in the MSI of a class, and show that a bound used for almost 15 years is incorrect. Applying the newly developed bounds on the MSI dimensions, we present an algorithm to enumerate all

equivalence classes with up to 100 boxes per pallet. Chapter IV presents new algorithms, both exact and heuristic, and bounding procedures for PLP.

Chapter V introduces MD-KP with the description of some previous work and discusses a MIP for 2D-KP. Chapter VI presents new algorithms for MD-KP, exact and heuristic, and compares the performance of these algorithms with others from the literature.

Chapter VII reviews previous work on MD-BPP, presents an exact algorithm for the orthogonal MD-BPP, and describes a new heuristic for 2D-BPP based on ideas presented in Chapters IV and VI.

Chapter VIII provides conclusions and suggestions for further research.

II. THE PALLET LOADING PROBLEM

This chapter reviews previous work on one of the simplest variations of 2D-KP, known as the Pallet Loading Problem (PLP). In this two-dimensional problem, all items, or *boxes*, to be packed have identical dimensions; there is only one bin, or *pallet*. Boxes may be rotated 90º and must be packed with edges parallel to the pallet's edges, i.e., the packing must be orthogonal. Using the classification of Dyckhoff [1990], PLP is classified as $2/B/O/C$.

Manufacturers must solve PLP when selecting the best arrangement of identical rectangular boxes, with a "this side up" restriction, on a rectangular pallet. Even without the "this side up" restriction, operational considerations may dictate the use of vertical layers with the same height. Considerations regarding stability and safety of the boxes imply the use of orthogonal packing, e.g., Dowsland [1987] and Nelissen [1995]. PLP is also present in some cutting stock and floor design settings.

Although simple when compared with 2D-KP, the complexity of PLP is still unknown. The one-dimensional version of this problem can be solved in constant time, but the same is not true for higher dimensions. The emphasis in most of the literature has been on finding good heuristic solutions. In the following sections, we present the problem formulation, define some concepts related to PLP, briefly review select PLP research, discuss the complexity of PLP, and survey bounding procedures and heuristics in the literature.

A. PROBLEM FORMULATION AND DEFINITIONS

1. PLP Problem Formulation

To be consistent with the PLP literature, we define some new notation. In each instance of PLP, identified by a quadruple (X, Y, a, b), we have a rectangular pallet with length X and width Y ($X \geq Y$), and a rectangular box with length a and width b ($a \geq b$). We can assume, without loss of generality, that X, Y, a, b are positive integers [Bischoff and Dowsland 1982]. We also assume that at least one box can be packed in the pallet. Thus, $X \geq a$ and $Y \geq b$.

The optimization version of PLP is: What is an orthogonal arrangement of the boxes on the pallet that yields the maximum number of boxes packed?

The decision version of PLP is: Given an instance of PLP, is there an orthogonal arrangement of N boxes on the pallet?

Some authors (e.g., Nelissen [1995], Scheithauer and Terno [1996]) formulate PLP in terms of the maximum number of boxes that can be packed on the pallet, without explicitly considering the arrangement of the boxes. The main difference between considering the arrangement or not is that it could be possible to develop an algorithm that computes the maximum number of boxes packed without defining the arrangement. If an arrangement is given, it is straightforward to calculate the number of boxes packed. On a planning level, we only need to know how many boxes can be loaded on each pallet, but at the operational level we must know how to arrange them. Because the complexity involved in solving PLP in either approach is presently unknown, we select the operational level.

2. PLP Definitions

We define an *H-box* (*V-box*) as a box with its largest dimension oriented horizontally (vertically). Given an instance (X',Y',a',b') of PLP, $N(X',Y',a',b')$ is the number of boxes in an optimal packing, $W(X',Y',a',b') \equiv X' * Y' - N(X',Y',a',b') * a' * b'$ is the wasted area in an optimum packing, $UN(X',Y',a',b')$ is an upper bound on $N(X',Y',a',b')$, and $EW(N',X',Y',a',b') \equiv X' * Y' - N' * a' * b'$ is the waste encountered when packing N' boxes for $N' \le \lfloor (X' * Y')/(a' * b') \rfloor$. Also $A_{x'} \equiv \lfloor X'/a' \rfloor$, $A_{y'} \equiv \lfloor Y'/a' \rfloor$, $B_{x'} \equiv \lfloor X'/b' \rfloor$, and $B_{y'} \equiv \lfloor Y'/b' \rfloor$.

We define a *block* as a rectangular subset of a packing arrangement such that no box is only partially included in the rectangle. In other words, a block partitions the boxes of a packing arrangement into two groups, those inside and those outside the block. If all boxes in a block have the same orientation (V-boxes or H-boxes), then the block is called a *homogeneous block* [Scheithauer and Terno 1996]. A homogeneous block of V-boxes (H-boxes) is a *V-block* (*H-block*).

A *hollow block* (Figure II.1) contains boxes in one orientation, across one diagonal of the block, surrounded by boxes in the other orientation. Each homogeneous block

located in the diagonal of the pattern is called a *diagonal element block*, or *diagonal element*. All boxes with a different orientation than those in the diagonal elements are located in *main element blocks*, or *main elements*. The unused regions inside the block are called *holes*. The hollow block is also called *diagonal block*.

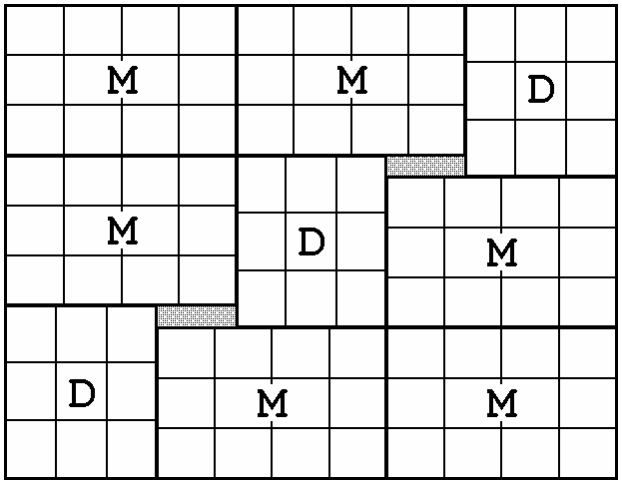

Figure II.1 *Feasible packing for the class of instance (85, 66, 8, 7).*
Diagonal elements are identified with the letter D. Main elements are identified with the letter M.

3. Efficient Partitions and Equivalence Classes

The idea of efficient partitions [Bischoff and Dowsland 1982] comes from the observation that some problems, with different dimensions, possess the same arrangement of the boxes in an optimal solution. The arrangement depicted in Figure II.2 is an optimal arrangement to the instance (22, 16, 5, 3). Shaded regions indicate unused (wasted) areas of the pallet. But the same arrangement is optimal to instances (30, 22, 7, 4) and (50, 36, 11, 7), among an infinite number of instances.

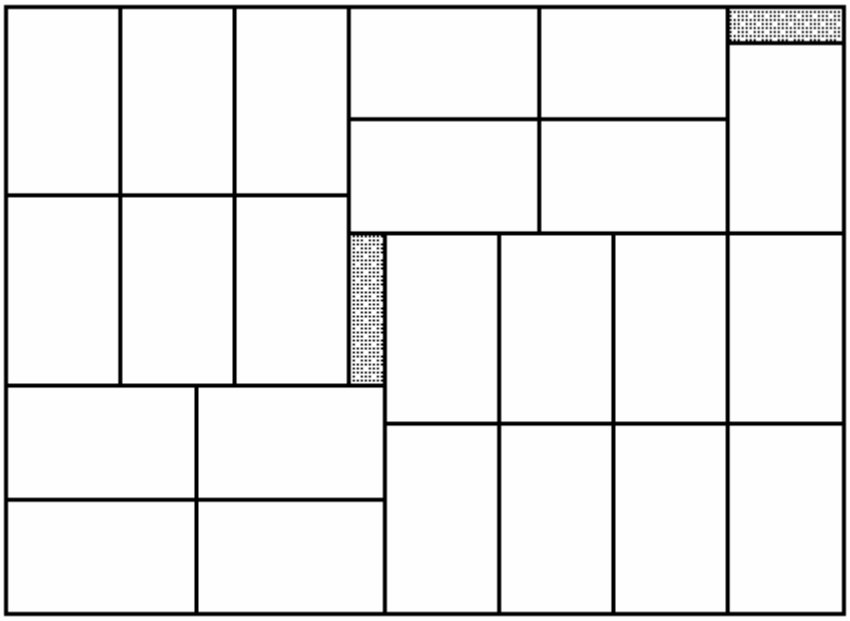

Figure II.2 *Optimal arrangement for instances (22, 16, 5, 3), (30, 22, 7, 4), (50, 36, 11, 7), and all instances within the same equivalence class.*

Now, if we draw a vertical or horizontal line across a loaded pallet, as in Figure II.3, it crosses a number of boxes. If the dimension of a box crossed by the line is *a*, we call the segment of the line contained within the box an *a-segment*. Otherwise, we call it a *b-segment*. The sum of the segments crossed by a vertical or horizontal line cannot exceed the related dimension of the pallet.

If we consider the possible combinations of *a*-segments and *b*-segments obtained for each dimension independently, we can define a set of inequalities to be satisfied for the arrangement to be feasible, i.e., a set of inequalities that ensure the combination of H-boxes and V-boxes does not exceed either the length or width of the pallet.

Let (n,m) denote an ordered pair of nonnegative integers satisfying

$$n * a + m * b \leq S \tag{II.1}$$

for a pallet dimension *S*, which could be *X* or *Y*. Then, the ordered pair (n,m) is called a *feasible partition* of *S*, i.e., a combination of segments of lengths a and b not exceeding the dimension *S* is a feasible partition of *S*. The set of all feasible partitions of the dimension *S* for boxes of dimensions $a \times b$ is denoted by $F(S,a,b)$.

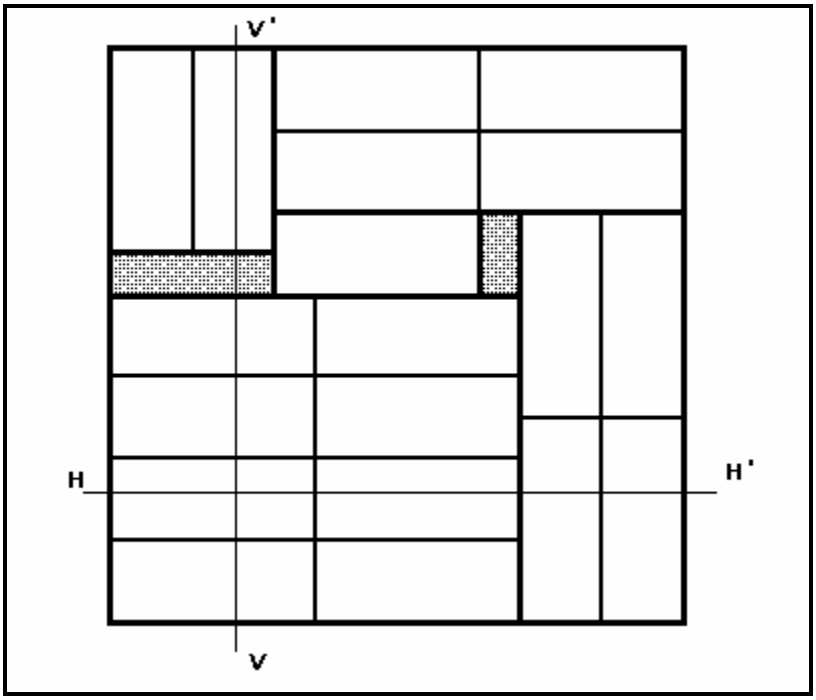

Figure II.3 *Example of identifying segments in a loaded pallet.*
The horizontal line $H-H'$ *crosses 4 boxes, two across the a dimension (two a-segments) and two across the b dimension (two b-segments). The vertical line* $V-V'$ *crosses 5 boxes, one across the a dimension and 4 across the b dimension.*

If n and m also satisfy

$$0 \le S - n*a - m*b < b \tag{II.2}$$

then (n,m) is called an *efficient partition* of S [Bischoff and Dowsland 1982]. We cannot increase an element of the ordered pair (n,m) in an efficient partition without becoming infeasible. For a pallet dimension S, the set of efficient partitions of S, denoted by $E(S,a,b)$, is defined to be the set of all feasible partitions (n,m) satisfying

$$n \in \{0,1,\ldots,\lfloor S/a \rfloor\}, \text{ and } m = \lfloor (S - n*a)/b \rfloor. \tag{II.3}$$

Dowsland [1984] shows that if two instances of PLP possess the same set of efficient partitions for both the pallet width and length, then both instances share the same set of optimal arrangements. This defines a relation in the set of instances of PLP, which is reflexive, symmetric, and transitive. Therefore, the set of instances of PLP can be divided into equivalence classes, based on the set of efficient partitions. If a solution is known for a class representative, then this solution can be used on any other instance in the class.

Because multiplying all dimensions by an integer produces a new instance in the same class, it is easy to see that each class contains infinitely many instances.

If, in addition, *n* and *m* satisfy

$$n * a + m * b = S, \tag{II.4}$$

then (*n*, *m*) is called a *perfect partition* of *S* [Dowsland 1984]. The set of all perfect partitions of *Y* is denoted $P(Y, a, b)$. There is a corresponding set for *X*. In general, each of these sets can be empty, but the instance of an equivalence class with minimal dimensions contains at least one perfect partition for each dimension, *X* and *Y* [Dowsland 1984]. This is easily observed if we consider an arbitrary instance without a perfect partition for a given dimension. In this case, we can reduce the corresponding dimension of the pallet without altering the set of efficient partitions. This implies that the new instance, with a smaller dimension, also belongs to the same class.

B. BACKGROUND ON PLP

In this section, we review some previous work on PLP; a substantial fraction of this work deals with developing heuristics and identifying upper bounds for PLP.

Barnett and Kynch [1967] are probably the first authors to publish a result regarding PLP. They consider the "problem of optimal dissection of a large rectangular plane area into smaller rectangles having unit width and integral length so as to obtain the least waste." Their main result is that if all boxes have equal integral lengths, then the maximum number of boxes packed can be computed in constant time. Barnes [1979] uses the idea that packing patterns of $a \times b$ boxes can be represented by patterns of $a \times 1$ or $b \times 1$ boxes, to compute a lower bound on the wasted area present in an optimal packing.

Brualdi and Foregger [1974] extends the result obtained by Barnett and Kynch [1967] to higher dimensions, and consider the case when the $a \times b$ box has harmonic dimensions, i.e., *b* divides *a*.

Dowsland [1987a] proposes an exact algorithm for PLP based on a graph-theoretic model of the problem. In her paper, she claims, without proof, that PLP is NP-complete. The same year, she investigates a combined database and algorithmic solution [1987b], and claims to have generated all equivalence classes satisfying a set of restrictions on the pallet

and box dimensions (enumerating some equivalence classes more than once). Dowsland [1990] also investigates the situation in which there are additional restrictions on loading patterns, as when robots are used in the loading process.

A working paper by Nelissen [Nelissen 1993] discusses the state of the art for PLP. More recently, Nelissen [1995] reports a bounding procedure based on a linear program (LP) and use of structural information about the problem.

Tarnowski et al [1994] propose a polynomial time algorithm for the guillotine cut version of PLP.

The G4-Heuristic [Scheithauer and Terno 1996] solves 99% of the randomly generated test instances they investigate, and solves all the instances in a set defined by Dowsland [1987a].

Bhattacharya et al [1998] propose a new algorithm that uses a *Maximal Breadth Filling Sequence*. This tree search algorithm is based on packing not individual boxes, but a combination of boxes corresponding to an efficient partition of a partial problem, at each step of the search. Although they claim the algorithm is exact, we provide a counter-example in the next chapter and show the errors in their proof.

C. COMPLEXITY OF PLP

PLP, regarded as 2D-KP, would be in the class of NP problems. In this case, the input would include a list with the repeated dimensions for each box. So, if we are trying to pack N boxes, the list has $2N$ values. But by "simplifying" the problem, we also simplified the input requirements. As seen above, every PLP instance can be defined using only four integers, and a "reasonable" encoding scheme would require $O(\log_2 X)$ as input size. If every solution *certificate* produced by an algorithm, i.e., the arrangement of the boxes on the pallet, is represented by the assignment of pairs of planar coordinates to each box, which may require $O((X/b)^2)$ values, it is easy to see that the certificate size is not bounded by a polynomial in the input size, and the problem would not even be a member of NP [Cook et al 1998, p. 310]. Nelissen [1993] reaches the same conclusion.

While the Guillotine PLP is solvable in $O(\log_2 X * \log_2^2 a)$ time [Tarnowski et al 1994], no similar result is available regarding the complexity of the non-guillotine case.

A polynomial-time algorithm for the non-guillotine PLP, if one exists, must encode the packing arrangement in a certificate with size bounded by a polynomial in $\log_2 X$. If it could be proven that no such encoding scheme exists, then it would be possible to show that the non-guillotine PLP is not in NP. On the other hand, the search for such an encoding scheme might lead to identifying a polynomial-time algorithm.

One possible approach for encoding the packing arrangement in a certificate with size bounded by a polynomial in the input size would be to represent the solution as a combination of blocks, including only boxes with the same orientation (only H-boxes or V-boxes), or other patterns that could be created in polynomial time. Some of the common heuristics for solving PLP involve the use of 1, 2, 4, or more blocks of boxes with the same orientation, or a special combination of blocks [Nelissen 1993]. In this case, the planar coordinates of each block, together with its dimensions, would define the solution arrangement. If the number of such blocks required to represent an optimal solution could be bounded by a polynomial in $\log_2 X$, then it would be possible, at least, to show that PLP is in NP. If it could be shown that the algorithm producing those blocks also had polynomial time complexity, than we could show PLP to be in the class P.

D. UPPER BOUNDS ON THE OPTIMAL SOLUTION

The existence of a tight upper bound on the maximum number of boxes that can be packed in an instance of PLP is useful both in heuristics, to verify optimality, and for exact algorithms, to enable bounding strategies that reduces the solution space. This dissertation introduces some new bounds in Chapter IV. In this section, we review some previously published bounds.

1. Simple Bounds

A simple and intuitive bound is obtained by taking the integer part of the ratio of the areas of pallet and box, $\lfloor (X * Y)/(a * b) \rfloor$. We call this the *Area Ratio (AR) Bound*.

Although simple to compute, Smith and De Cani [1980] and Dowsland [1985] report that it is equal to the optimal solution in less than 15% of randomly generated test instances.

Another intuitive bound is obtained by multiplying B_x, the maximum number of boxes that can be placed in any horizontal row, and B_y, the maximum number of boxes in a vertical column, obtaining $\lfloor X/b \rfloor * \lfloor Y/b \rfloor$. If *b* is relatively close to *a*, then this *Maximum Product (MP) Bound* can outperform the AR bound. The instance (23, 23, 5, 4) is an example of this situation, with an AR bound $\lfloor (23*23)/(5*4) \rfloor = 26$ and MP bound $\lfloor 23/4 \rfloor * \lfloor 23/4 \rfloor = 25$.

2. Bound Using a Perfect Partition Equivalent of the Pallet

If an instance of PLP presents perfect partitions in both dimensions, the reduction of the pallet dimensions to the largest efficient partition can significantly improve the performance of the area ratio bound [Dowsland 1984]. In this situation, the dimensions of the pallet are not a positive linear combination of the box's dimensions, and there is always some wasted area in each dimension of the pallet. If we compute the minimum waste observed in each pallet dimension, and reduce the dimensions accordingly, the new bound is much tighter, being "correct for over 70% of common box and pallet dimensions" [Dowsland 1984].

Let $G(S,\hat{a},\hat{b}) = \max_{(i,j)\in E(S,\hat{a},\hat{b})} \{i*\hat{a} + j*\hat{b}\}$. We call $G(S,\hat{a},\hat{b})$ the *Perfect Partition Equivalent* function. Given an instance $(\hat{X},\hat{Y},\hat{a},\hat{b})$ of PLP, the reduced dimensions of the pallet are given by

$$X^* = G(\hat{X},\hat{a},\hat{b}), \tag{II.5}$$

and

$$Y^* = G(\hat{Y},\hat{a},\hat{b}). \tag{II.6}$$

For instance (38, 38, 12, 5), the area ratio bound computed directly is 24. After reducing to the usable dimensions to $X^* = 37$ and $Y^* = 37$, we obtain the instance (37, 37, 12, 5), with an exact bound of 22.

3. Minimizing the Area Ratio Bound on the Equivalence Class

Selecting an instance in the equivalence class that minimizes the area ratio bound may provide a tighter bound. Because all instances in an equivalence class can be packed in the same way, with the same optimum number of packed boxes, the bound computed for one instance is valid for all instances. Proposed by Dowsland [1984], this approach produces exact bounds in at least 92% of her 5,000 randomly generated test cases. We call this bound the *Minimum Area Ratio (MAR) Bound*.

The selection of an instance that minimizes the area ratio bound can be stated as a nonlinear integer-programming problem. Given instance $(\hat{X},\hat{Y},\hat{a},\hat{b})$, we formulate the minimization problem as

$$Min\lfloor (X*Y)/(a*b)\rfloor$$

subject to

$$X - i*a - j*b \geq 0, \qquad \forall (i,j) \in E(\hat{X},\hat{a},\hat{b}), \tag{II.7}$$

$$i*a + (j+1)*b - X > 0, \qquad \forall (i,j) \in E(\hat{X},\hat{a},\hat{b}), \tag{II.8}$$

$$(\lfloor \hat{X}/\hat{a} \rfloor + 1)*a - X > 0, \tag{II.9}$$

$$Y - f*a - g*b \geq 0, \qquad \forall (f,g) \in E(\hat{Y},\hat{a},\hat{b}), \tag{II.10}$$

$$f*a + (g+1)*b - Y > 0, \qquad \forall (f,g) \in E(\hat{Y},\hat{a},\hat{b}), \tag{II.11}$$

$$(\lfloor \hat{Y}/\hat{a} \rfloor + 1)*a - Y > 0, \tag{II.12}$$

$$X,Y,a,b \in Z^{+}.$$

The nonlinear objective function, together with integrality constraints on variables, can make this minimization problem hard to solve, but Dowsland [1987] and Nelissen [1993] show how to convert this problem to a simpler one.

The first step in the conversion is to allow the variables to take continuous, nonnegative values, and to consider only a normalized version of the problem with the box having width *b* always equal to 1. If the optimal solution has fractional rational values, we can scale up to obtain only integral values. If decision variables in the optimal solution are irrational, we can obtain a rational solution that is arbitrarily close [Nelissen 1993].

The second step is to recognize that the length and width of the pallet, in the optimal solution, is given by a linear function of the box dimensions, with the coefficients given by a pair of perfect partitions. This way, given a pair of perfect partitions for length and width, say (i^*, j^*) and $\left(f^*, g^*\right)$, the objective function can be rewritten as a function of the box length only:

$$AR(a) = \frac{(i^* * a + j^*) * (f^* * a + g^*)}{a} = \frac{j^* * g^*}{a} + i^* * f^* * a + (i^* * g^* + j^* * f^*). \quad \text{(II.13)}$$

The first and second derivatives are, respectively,

$$AR'(a) = -\frac{j^* * g^*}{a^2} + i^* * f^*, \text{ and} \quad \text{(II.14)}$$

$$AR''(a) = \frac{2 * j^* * g^*}{a^3}. \quad \text{(II.15)}$$

Given a pair of efficient partitions, $AR(a)$ is a convex function for positive values of a, and the function attains the minimum value at an interior point where $AR'(a) = 0$, or at the boundary of the feasible region. As Nelissen [1993] demonstrates, the domain of a can be divided into intervals where one pair of efficient partitions dominates all others. As we increase the value of a from its minimum value of one, we move from one pair of efficient partitions to another. The algorithm verifies each interval sequentially until the instance with minimum area ratio is identified. The feasible region has an open boundary, and if the objective function attains a minimum value at this open boundary, we can produce an instance with area ratio arbitrarily close to the minimum value, but still belonging to the equivalence class.

One example of a class where the objective function attains a minimum value at the open boundary is the class of instance $(57,44,12,5)$. This instance has $AR = 41.8$. When minimizing over the equivalence class, the minimum is obtained at instance $(23,18,5,2)$, with $AR = 41.4$. But the Y-partition $(0,9)$ is feasible for $(23,18,5,2)$, and infeasible for $(57,44,12,5)$. Therefore, $(23,18,5,2)$ belongs to a different class, but instance $(23057,18044,5012,2005)$ belongs to the same class as $(57,44,12,5)$, and has $AR = 41.401$.

4. Barnes' Bound

Barnes [1979] proposes a bound based on patterns with $a \times b$ boxes being represented by patterns of $a \times 1$ or $b \times 1$ boxes. Because the solution to problems with unit width is easily obtained [Barnett and Kynch 1967], Barnes computes the wasted area obtained when packing only $a \times 1$ or only $b \times 1$ boxes, and uses the greater wasted area as a lower bound on the wasted area of the original problem, producing an upper bound on the number of boxes packed. Barnes also proves that if the dimensions of the pallet are "sufficiently large" then the bound is exact.

If A is the wasted area in an optimal packing of a × 1 boxes, $r_a = X \bmod a$ and $s_a = Y \bmod a$, then

$$A = \min\{r_a * s_a, (a - r_a) * (a - s_a)\}. \tag{II.16}$$

The wasted area in an optimal packing of b × 1 boxes, B, is computed in a similar way. In any optimal packing of a × b boxes, the wasted area, $W(X, Y, a, b)$, satisfies the following system of equations [Barnes 1979]

$$W(X, Y, a, b) \geq \max\{A, B\}, \tag{II.17}$$

$$W(X, Y, a, b) \bmod a = A \bmod a, \text{ and} \tag{II.18}$$

$$W(X, Y, a, b) \bmod b = B \bmod b. \tag{II.19}$$

An example of the application of this bound is the instance $(22, 18, 4, 3)$. The area ratio bound is $\lfloor (22 * 18)/(4 * 3) \rfloor = 33$, with zero wasted area. The calculations required to apply Barnes' bound are

$$r_a = 2,\ s_a = 2, \qquad A = \min\{4, 4\} = 4,$$

$$r_b = 1,\ s_b = 0, \qquad B = \min\{0, 6\} = 0,$$

$$W(22, 18, 4, 3) \geq \max\{4, 0\} = 4,$$

$$W(22, 18, 4, 3) \bmod 4 = 4 \bmod 4, \text{ and}$$

$$W(22, 18, 4, 3) \bmod 3 = 0 \bmod 3.$$

The minimum value satisfying the system of equations is 12. Therefore, the bound can be reduced to $((22 * 18) - 12)/(4 * 3) = 32$.

5. An LP Bound

Nelissen [1995] reports that in 1987 Isermann presented a method for computing upper bounds on the number of boxes in the optimal packing by solving an LP. In this approach, the pallet is partitioned into vertical and horizontal strips of unit width. The horizontal strips are called *rows*, and the vertical strips *columns*. Given a normal packing pattern (left-bottom justified, as defined in Chapter I) corresponding to an instance (*X*, *Y*, *a*, *b*) of PLP, the number of a and b segments in each strip is recorded as an ordered pair. Figure II.4 depicts an optimal arrangement for instance $(8,5,3,2)$ (after Nelissen [1995]) that demonstrates the use of this notation. For each pair of integers corresponding to each row and column, the first integer indicates the number of *a*-segments in that unit strip. The second integer is the number of *b*-segments.

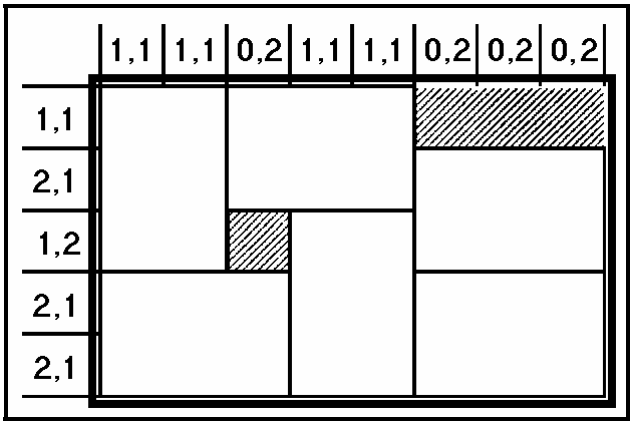

Figure II.4 *Partitions observed on a packing of instance (8, 5, 3, 2).*
For each pair of integers corresponding to each row and column, the first integer indicates the number of a-segments in that unit strip. The second integer is the number of b-segments. As an example, the row assigned with (1,2) contains one a-segment and two b-segments. Figure after Nelissen [1995].

Let the integer variable $x_{i,j}$ count the number of times the feasible partition (i, j) of X is observed in a solution. In our example above, $x_{1,1} = 1$, $x_{1,2} = 1$, $x_{2,1} = 3$, and $x_{i,j} = 0$ for all other partitions in $F(8,3,2)$. Y partitions are counted in $y_{f,g}$. In this example, $y_{0,2} = 4$, $y_{1,1} = 4$, and $y_{f,g} = 0$, for all other partitions in $F(5,3,2)$.

Each V-box in the solution is counted in a rows and in b columns, while each H-box is counted in b rows and a columns. Any given 1×1 square, aligned with the grid and occupied by a box, is counted twice, once in the row and another in the column. The

product $a \times i \times x_{i,j}$ gives us the number of 1×1 squares belonging to H-boxes counted in rows with i H-boxes and j V-boxes. If we add over all feasible partitions, we obtain the total number of unit squares, or total area, occupied by H-boxes in the packing. If we divide by the area of an item, $a * b$, we obtain the total number of H-boxes in the packing. Also, the product $b * g * y_{f,g}$ represents the number of 1×1 squares of H-boxes in a column with f V-boxes and g H-boxes, and the sum over all feasible partitions also returns the area occupied by H-boxes. Therefore, the result of both summations must be the same. A similar reasoning applies to the V-boxes. Additionally, we relax the integrality constraint on $x_{i,j}$ and $y_{f,g}$ and then formulate the problem as

$$Max\ Z = H + V$$

subject to

$$H = \sum_{(i,j) \in F(X,a,b)} \frac{i * x_{i,j}}{b}, \tag{II.20}$$

$$V = \sum_{(f,g) \in F(Y,a,b)} \frac{f * y_{f,g}}{b}, \tag{II.21}$$

$$\sum_{(i,j) \in F(X,a,b)} x_{i,j} = Y, \tag{II.22}$$

$$\sum_{(f,g) \in F(Y,a,b)} y_{f,g} = X, \tag{II.23}$$

$$\sum_{(i,j) \in F(X,a,b)} a * i * x_{i,j} - \sum_{(f,g) \in F(Y,a,b)} b * g * y_{f,g} = 0, \tag{II.24}$$

$$\sum_{(i,j) \in F(X,a,b)} b * j * x_{i,j} - \sum_{(f,g) \in F(Y,a,b)} a * f * y_{f,g} = 0, \tag{II.25}$$

$$x_{i,j} \geq 0, \quad \forall (i,j) \in F(X,a,b), \text{ and} \tag{II.26}$$

$$y_{f,g} \geq 0, \quad \forall (f,g) \in F(Y,a,b). \tag{II.27}$$

The integer part of *H*, as defined in (II.20), counts the number of H-boxes, while the integer part of *V*, in (II.21), counts the number of V-boxes. Constraints (II.22) and (II.23) require that the number of horizontal and vertical strips be equal to the vertical and horizontal dimensions, respectively. Constraints (II.24) and (II.25) enforce the area occupied by H-boxes and V-boxes to be the same, as discussed above. The integer part of *Z* is an upper bound on the number of boxes that can be packed in the pallet.

Nelissen [1995] and Naujoks (as reported by Nelissen [1995]) study and develop improvements to this method, with Nelissen reporting that his proposed procedure yields the optimal number of boxes packed in 99.86% of 20,000 randomly generated test instances with at most 50 boxes, and 99.81% in an additional 20,000 problems, with between 51 and 100 boxes.

As we shall see in the next chapter, this record is surpassed by a combination of new bounds developed in this dissertation.

E. SIMPLE HEURISTICS FOR PLP

Nelissen [1993] reviews the literature on PLP and furnishes most references in this section.

Most heuristics developed so far are based on the idea of combining V-blocks and H-blocks. As the possible number of blocks in the solution increases, better solutions can be obtained with the development of more complex heuristics, at additional computational cost. A tight upper bound on the number of blocks necessary in the optimal solution for every PLP is still unknown [Scheithauer and Terno 1996].

Some authors use a sample set of randomly generated test problems to measure the performance of their algorithms, usually observing some set of restrictions on pallet and box dimensions. The most common set of restrictions is first proposed by Dowsland [1984] and has been used by other authors, e.g., Nelissen [1993], Scheithauer and Terno [1996], Bhattacharya et al [1998], Morabito and Morales [1998]. These restrictions are

$$1 \le X/Y \le 2\,, \tag{II.28}$$

$$1 \le a/b \le 4\,, \text{ and} \tag{II.29}$$

$$1 \le (X * Y)/(a * b) \le 50\,. \tag{II.30}$$

Nelissen [1995] and Naujoks (as reported by Nelissen [1995]) also investigate instances with the area ratio in the range

$$51 \le (X * Y)/(a * b) \le 100\,. \tag{II.31}$$

We enumerate all equivalence classes with AR bound up to 100 boxes, and to solve one instance in each class.

1. One-Block Heuristic

In the *one-block heuristic*, all boxes are packed with the same orientation. The decision on which orientation to use is based on constant-time computations. If $A_x * B_y > A_y * B_x$, then an H-block with $A_x * B_y$ H-boxes is used, divided into A_x columns and B_y rows. Otherwise, a V-block with $A_y * B_x$ V-boxes is used, with B_x columns and A_y rows of boxes.

Although it is a very simple heuristic, we find that it optimally solves more than 58% of all problems satisfying restriction (II.31). An optimal arrangement obtained with the one-block heuristic is depicted in Figure II.5.

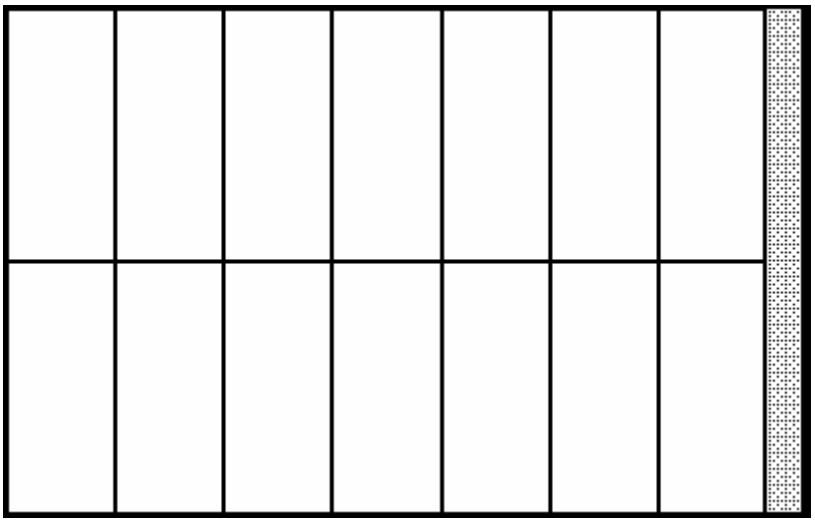

Figure II.5 *Optimal arrangement for the class of instance (22, 14, 7, 3) provided by the one-block heuristic.*
In this example, $A_x = 3, A_y = 2, B_x = 7, and\ B_y = 4$.

2. Two-Block and Three-Block Heuristics

All *n*-block heuristics, for $n \geq 2$, are based on selecting a size for the first block, then for the second block, until the last block is packed. If unused regions exist, then the heuristic tries packing in these regions.

In the *two-block heuristic*, for each partition (n,m) in $E(X,a,b)$, we generate two blocks: one H-block with *n* columns and B_y rows, and one V-block with *m* columns and A_y rows. We also generate two blocks, in a similar way, for each partition (n,m) in $E(Y,a,b)$, and pick the best solution, i.e., the solution with the most packed boxes. Figure

II.6 shows an optimal arrangement for the class of the instance $(21,11,4,3)$ obtained with the two-block heuristic.

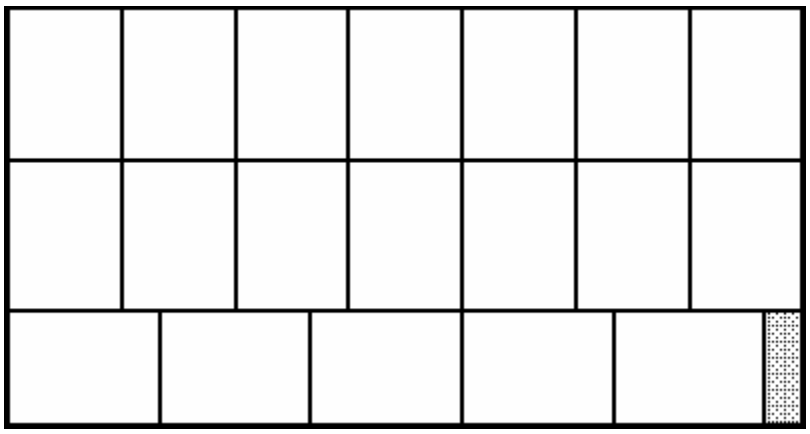

Figure II.6 *Optimal arrangement for the class of instance (21, 11, 4, 3) obtained with the two-block heuristic.*
This pattern corresponds to the efficient partition (2, 1) in $E(11,4,3)$*, with* $A_x = 5$ *and* $B_x = 7$.

In the *three-block heuristic*, after computing the results for each efficient partition, if there is an unused region that is large enough to fit another block, then the boxes in this third block are added to the solution corresponding to that partition, before selecting the best solution. Figure II.7 is an example of an optimal arrangement provided by the three-block heuristic.

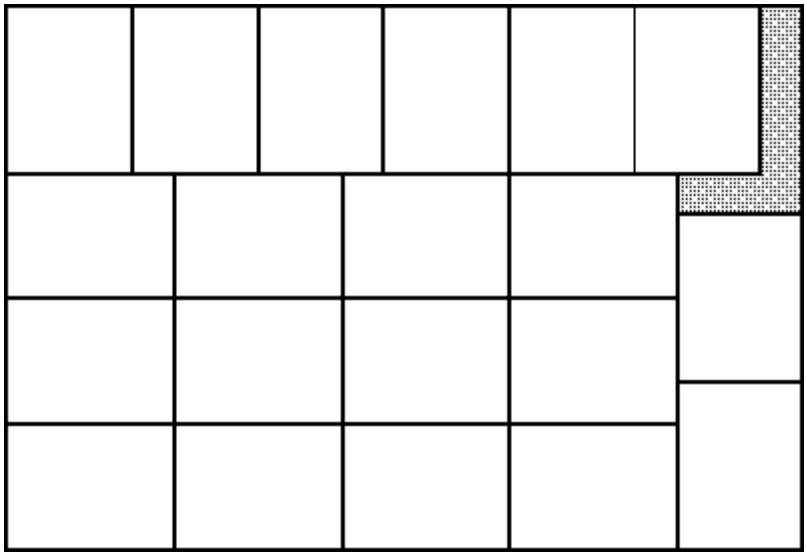

Figure II.7 *Optimal arrangement for the class of instance (19, 13, 4, 3) computed with the three-block heuristic.*
A third block with two V-boxes is added to the initial solution of the two-block heuristic.

3. Four-Block and Five-Block Heuristics

The *four-block heuristic*, proposed by Steudel [1979] and Smith and de Cani [1980], allocates a block with fixed orientation to each of the four corners of the pallet, as shown in Figure II.8. Blocks *I* and *III* are H-blocks, while blocks *II* and *IV* are V-blocks.

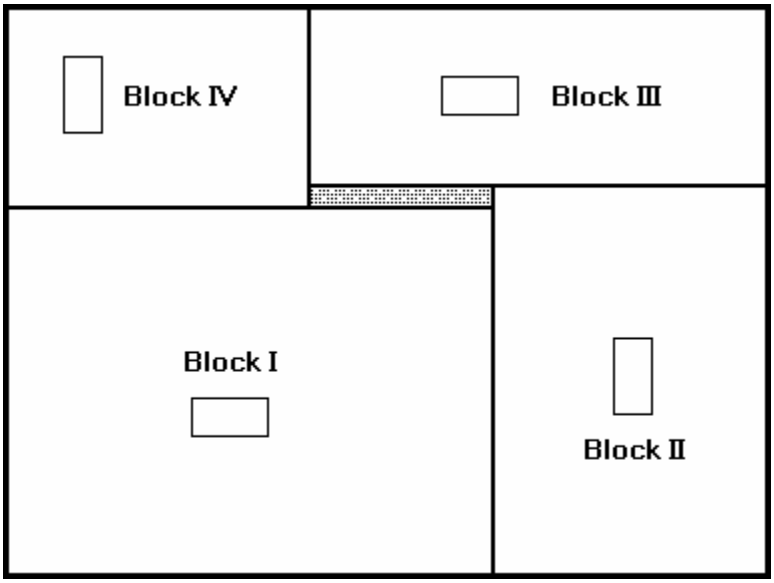

Figure II.8 *Orientation of items within blocks in the four-block heuristic. Blocks I and III are H-blocks, and blocks II and IV are V-blocks.*

Bischoff and Dowsland [1982] propose an algorithm using four nested loops, based on efficient partitions, to compute the dimensions of the four blocks. If there is still some unused area after picking dimensions of the four blocks, the algorithm tries to fit a fifth block with the best orientation for the specific instance. This is the *five-block heuristic*.

Let (L_i, W_i) be the dimensions, and $C_i = L_i * W_i$, the area of the i^{th} block, for $i = 1, 2, 3, 4, 5$. The pseudocode for the five-block heuristic is:

For each $(n_1, m_1) \in E(X, a, b)$, $L_1 \leftarrow n_1 * a, L_2 \leftarrow m_1 * b$

  For each $(n_2, m_2) \in E(Y, a, b)$, $W_1 \leftarrow n_2 * b, W_4 \leftarrow m_2 * a, C_1 \leftarrow L_1 * W_1$

    For each $(n_3, m_3) \in E(X, a, b)$, $L_3 \leftarrow n_3 * a, L_4 \leftarrow m_3 * b, C_4 \leftarrow L_4 * W_4$

      For each $(n_4, m_4) \in E(Y, a, b)$, $W_3 \leftarrow n_4 * b, W_2 \leftarrow m_4 * a, C_3 \leftarrow L_3 * W_3, C_2 \leftarrow L_2 * W_2$

        If no blocks overlap, then

          Compute L_5, W_5, and C_5

          If $C_1 + C_2 + C_3 + C_4 + C_5$ is larger than the previous best, save this solution.

Verifying that there is an overlap among the four initial blocks and computing the dimensions of the fifth block can be performed at the same time, because both answers depend on the same set of conditions:

If $L_1 + L_3 > X$ then

If $W_1 + W_3 > Y$, then there is overlap

else $W_5 = Y - (W_1 + W_3)$, and $L_5 = X - (L_2 + L_4)$

else if $L_2 + L_4 > X$ then

If $W_2 + W_4 > Y$, then there is overlap

else $W_5 = Y - (W_2 + W_4)$, and $L_5 = X - (L_1 + L_3)$.

The final step in this heuristic computes the *usable area* C_5 of the fifth, i.e. central, block. This is performed by applying the one-block heuristic to instance (L_5, W_5, a, b). Figure II.9 displays an optimal arrangement to the class of instance (14, 14, 5, 2) obtained with the five-block heuristic.

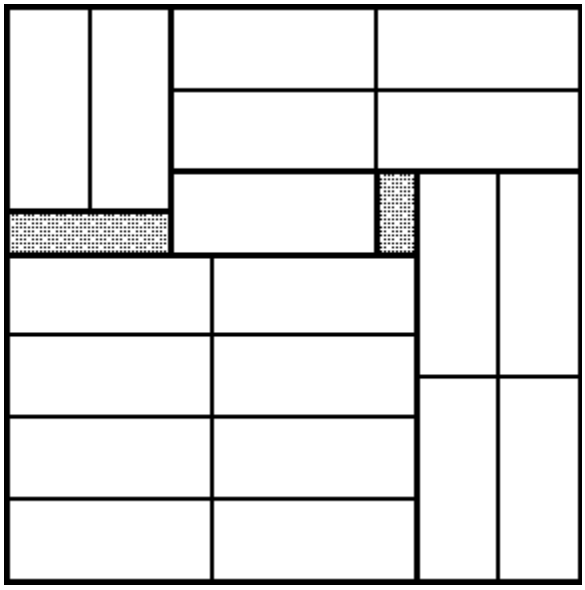

Figure II.9 *Optimal arrangement obtained with the five-block heuristic for the class of instance (14, 14, 5, 2).*

4. Diagonal and Angle Block Heuristics

Nelissen [1993] describes a family of *diagonal heuristics*, which generate diagonal block patterns, as in Figure II.10.

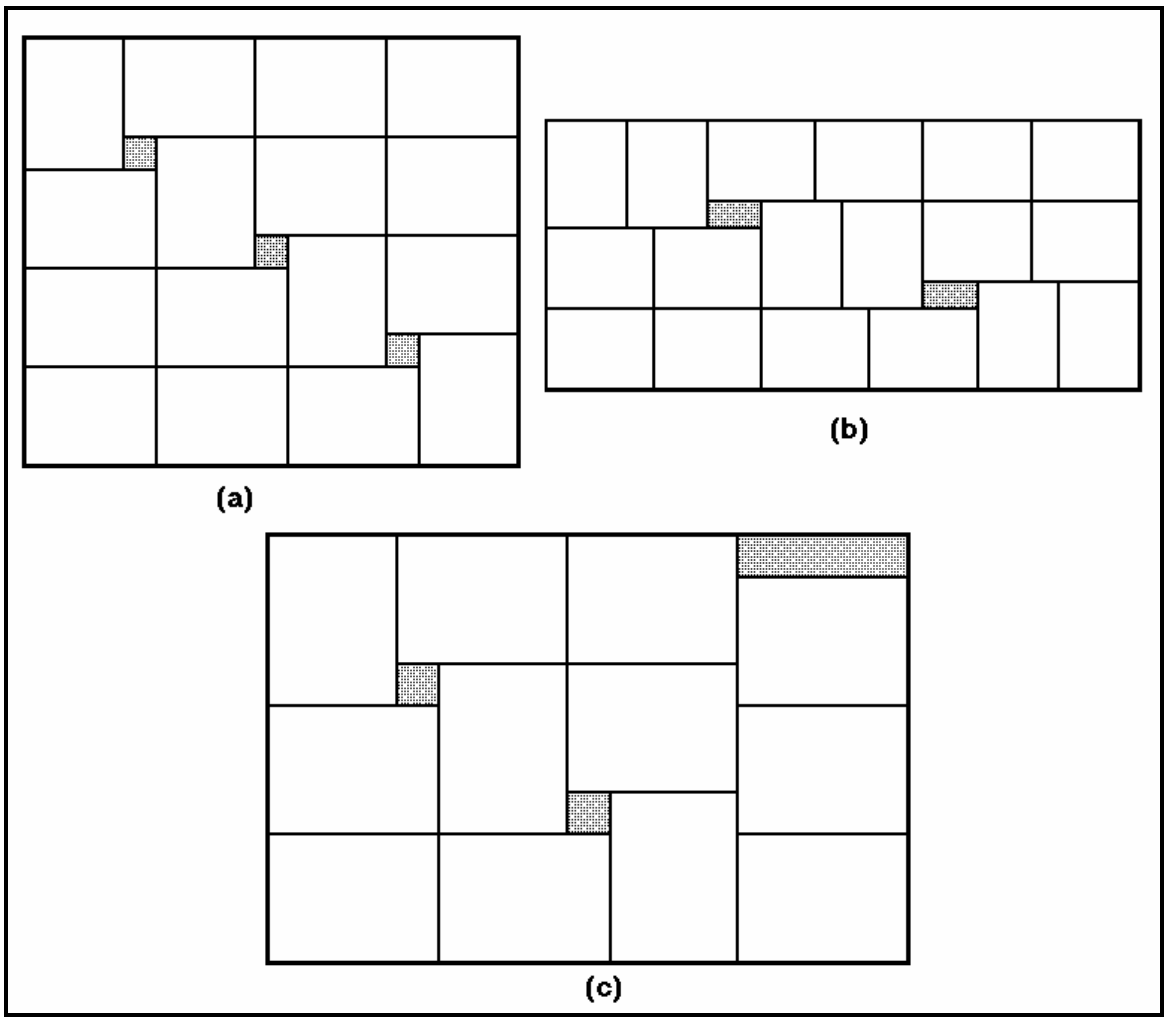

Figure II.10 *Examples of optimal packing patterns generated with diagonal block heuristics.*
(a) is a simple diagonal solution for the class of instance (15, 13, 4, 3); (b) is a multiple diagonal solution for the class of instance (22, 10, 4, 3); and (c) is a supplemented diagonal solution for the class of instance (15, 10, 4, 3).

If the solution to a problem has only one box in each diagonal element, he calls the pattern a *simple diagonal block solution*. If the diagonal elements are composed of more than one box, side by side, the solution is called a *multiple diagonal solution*, as in Figure II.10 (b). If the arrangement with diagonal block pattern is complemented by an additional block, as in Figure II.10 (c), it is called a *supplemented diagonal solution*. Nelissen [1993]

reports the work of other researchers with this heuristic. An important characteristic of the diagonal block pattern is its symmetry, with all holes having the same size.

Nelissen [1993] proposes other heuristics, using the same basic idea, the *angle heuristic* and the *recursive angle heuristic*, which allow more complex arrangements than the diagonal heuristics. The main difference between the diagonal heuristics and the angle heuristics is that the arrangement obtained with these latter procedures may not be symmetric, and holes can have different sizes, as shown in Figure II.11.

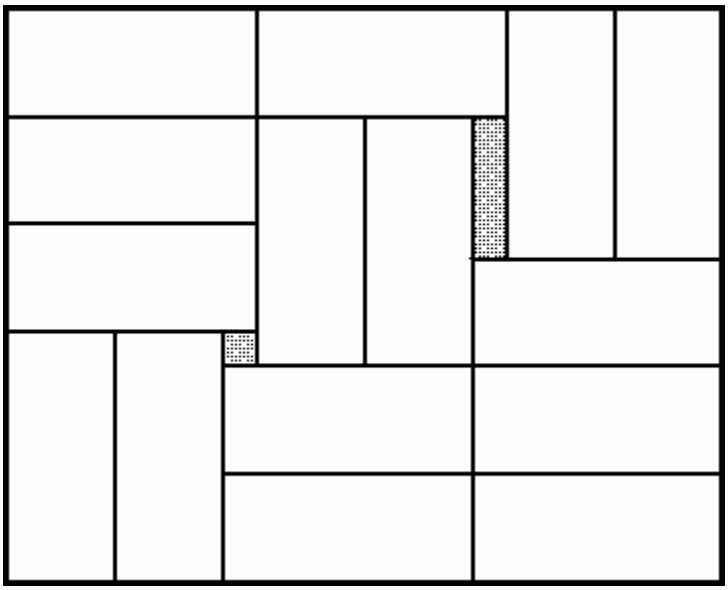

Figure II.11 *Optimal packing pattern obtained with the angle heuristic for the class of instance (20, 16, 7, 3).*

The angle heuristic uses three nested loops. The outer loop determines if additional boxes can be packed in the pallet, and there is one loop for each dimension. At each step of the procedure, a fraction of the current length and width is packed with a partial solution computed from the efficient partitions, and three new homogeneous blocks are added to the current solution: one at the left lower corner, one to the right, and one above the first block. If (i, j) is an efficient partition of the current length, (f, g) is an efficient partition of the current width, and $i, j, f, g > 0$, then four possible arrangements are analyzed, depending on the selection of the orientation of the boxes for each block, and the relative size of these blocks:

- A $j \times f$ block of V-boxes at the lower left corner of the current partial pallet, with an $i \times k$ block of H-boxes to the right, and a $l \times g$ block of H-boxes above,

where in the first arrangement $(k,l) = (\lceil f*a/b \rceil, \lfloor j*b/a \rfloor)$ and in the second arrangement $(k,l) = (\lfloor f*a/b \rfloor, \lceil j*b/a \rceil)$, and

- An $i \times g$ block of H-boxes at the lower left corner of the current partial pallet, with a $j \times k$ block of V-boxes to the right, and a $l \times f$ block of V-boxes above, where in this third arrangement $(k,l) = (\lceil g*b/a \rceil, \lfloor i*a/b \rfloor)$ and in the forth $(k,l) = (\lfloor g*b/a \rfloor, \lceil i*a/b \rceil)$.

The selection of values for *k* and *l* determines whether the corner block is longer than the block above it, or "higher" than the block at the right. Figure II.12 (after Nelissen [1993]) pictures the four possible cases, for instance (40, 32, 7, 3), X-partition (4, 4) and Y-partition (2, 6).

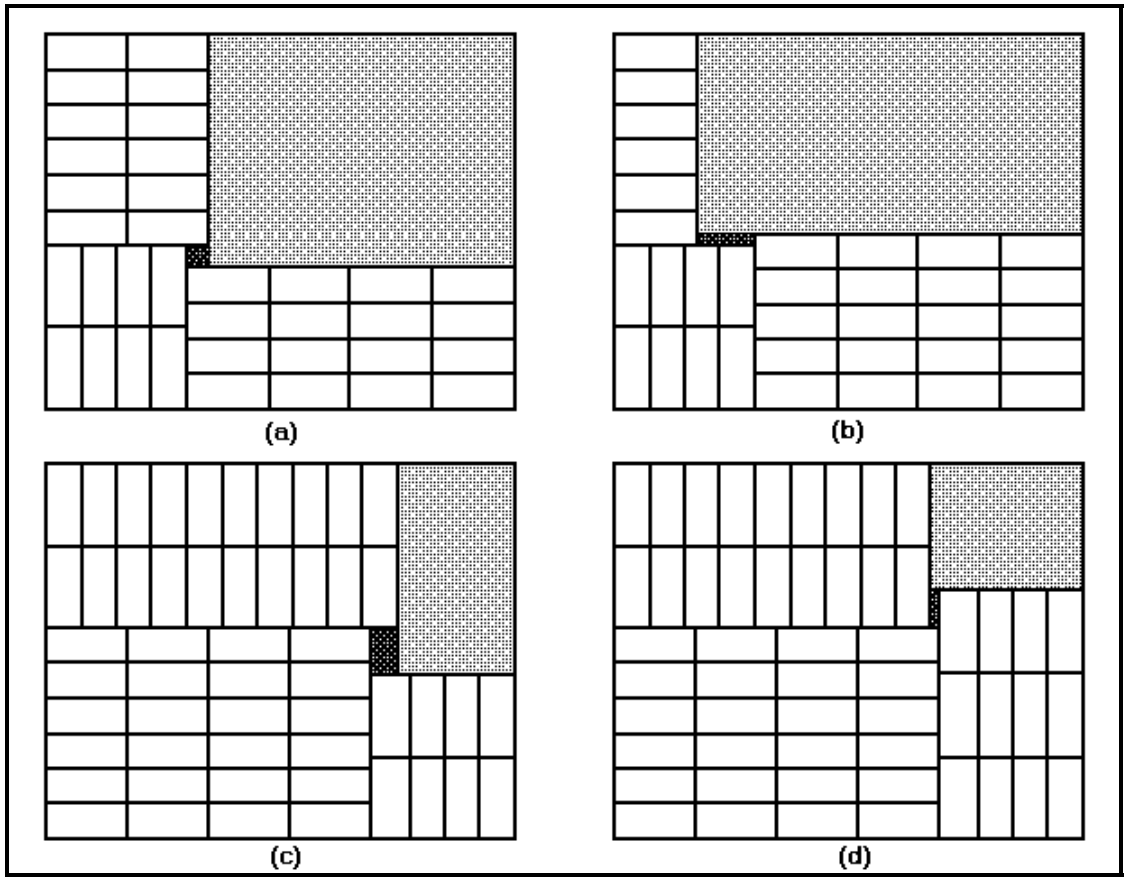

Figure II.12 *Initial partial solutions of the angle heuristic for the class of instance (40, 32, 7, 3).*

(a) solution with a block of V-boxes in the corner, $k = 4$ and $l = 2$; (b) also a corner block with V-boxes, but $k = 5$ and $l = 1$; (c) a block with H-boxes in the corner, $k = 2$ and $l = 10$; (d) also a corner block with H-boxes, but $k = 3$ and $l = 9$ (after Nelissen [1993]).

In the angle heuristic, the *angle*, or selection of the type of block in the corner, and values for k and l are fixed, and are repeatedly applied in the following steps of the procedure. If a used region remains in the pallet, the one-block heuristic is used to finish up the packing pattern.

In the recursive angle heuristic, a different angle is applied at each step of the procedure, allowing the creation of more complex packing arrangements.

F. MORE COMPLEX HEURISTICS

When simple heuristics fail to pack a number of boxes equal to the best available upper bound, more complex heuristics may be useful. Different types of complex heuristics, usually based on recursion, are available in the literature.

Nelissen [1993] proposes extending the recursive angle heuristic, allowing the use of a simple heuristic at each step of the procedure. In this case, at every step, the procedure tries a different angle and tests if a five-block heuristic yields a better solution. Nelissen also includes the possibility of completing the solution generated with the angle heuristic with one or more homogeneous blocks. This heuristic is named the *recursion heuristic*. The same paper proposes another heuristic, the *complex-block heuristic*. This heuristic is based on a seven-block heuristic, originally formed with homogeneous blocks, but it allows the central block to be an angle block, or a block generated by another heuristic.

Scheithauer and Terno [1996] propose the *G4-heuristic*. This heuristic is a recursive four-block heuristic. The G4-heuristic is able to solve all problem instances defined by conditions (II.28) to (II.30), and able to solve approximately 99% of their test instances.

Morabito and Morales [1998] propose another application of recursion to a simple heuristic. Their algorithm is a recursive five-block heuristic, based on the procedure proposed by Bischoff and Dowsland [1982].

THIS PAGE INTENTIONALLY LEFT BLANK

III. EQUIVALENCE CLASSES IN PLP

This chapter discusses PLP equivalence classes and analyzes new relationships among instances of PLP.

A. REPRESENTING EQUIVALENCE CLASSES

Because instances of PLP in the same equivalence class share the same set of optimal solutions, once one instance is solved, the solution can be stored in a database and retrieved whenever a solution to an instance of PLP of the same class is necessary [Dowsland 1987a]. The most straightforward way to identify each equivalence class in a database is to encode the set of efficient partitions defining the class. This way, given a new instance, it is possible to compute the set of efficient partitions and compare it with the entries in the database. One possible problem is that the cardinality of this set increases with the number of boxes packed.

Another approach is to select a unique class representative. This way, only four integers are necessary to represent the class, independent of the number of boxes in the optimum packing. But identification of instances belonging to the same class must be performed efficiently.

One option for defining an equivalence-class representative is the instance that minimizes the area ratio bound, the *Minimum Area Ratio Instance* (*MARI)*. But the minimization problem can have a solution at an open boundary, or at a non-integral interior point, as discussed in Chapter II, and in these cases the dimensions of the MARI can only be approximated, when using integers. Different approximations can generate different instances within the same class, complicating the identification process.

Another candidate for equivalence-class representative is the Minimum Size Instance (MSI), the instance that minimizes the dimensions of both the pallet and the box. We say that instance $(\tilde{X},\tilde{Y},\tilde{a},\tilde{b})$ is the *Minimum Size Instance* of a class if for all instances (X,Y,a,b) in the same class, $\tilde{X} \le X, \tilde{Y} \le Y, \tilde{a} \le a, \tilde{b} \le b$.

1. Existence and Uniqueness of the Minimum Size Instance

As explained when presenting the Perfect Partition Equivalent Bound, in Chapter II, given an instance $(\hat{X},\hat{Y},\hat{a},\hat{b})$ of PLP we can obtain the pallet with minimum allowed dimensions, $X^* \times Y^*$, for the given box dimensions, applying the perfect partition equivalent function. Therefore, if the dimensions of the box in the MSI, $\tilde{a} \times \tilde{b}$, are known, then the dimensions of the pallet are given by $\tilde{X} = G(\hat{X},\tilde{a},\tilde{b})$ and $\tilde{Y} = G(\hat{Y},\tilde{a},\tilde{b})$. Therefore, the dimensions of the pallet in the MSI of a given equivalence class are determined once the dimensions of the box are known.

We show that the MSI is unique in a class and its dimensions can be easily bounded, simplifying the process of enumerating equivalence classes.

Theorem III.1: Every equivalence class of PLP has one and only one MSI.

Proof: We initially show that there is no more than one MSI in each class. Then we show that every class has at least one MSI.

Suppose (X_1,Y_1,a_1,b_1) and (X_2,Y_2,a_2,b_2) are two MSIs in an equivalence class. By definition, both instances minimize all dimensions of the pallet and the box ($X_1 \le X_2, Y_1 \le Y_2, a_1 \le a_2, a_1 \le a_2$ and $X_2 \le X_1, Y_2 \le Y_1, a_2 \le a_1, a_2 \le a_1$), implying $X_1 = X_2, Y_1 = Y_2, a_1 = a_2, a_1 = a_2$. Therefore, if there is a MSI, it is unique.

Now consider an equivalence class. Because the dimensions of the pallet in the MSI are a function of the dimensions of the box in the MSI, the only way for a class not to have an MSI is if there exists one instance, say $(X_1,Y_1,\tilde{a},b_1)$, with minimum length for the box (i.e., $\tilde{a} \le a$ for all instances (X,Y,a,b) in the same class) and another instance, $(X_2,Y_2,a_2,\tilde{b})$, in which the box has minimum width (i.e., $\tilde{b} \le b$ for all instances (X,Y,a,b) in the class). In this case, $a_2 > \tilde{a}$ and $b_1 > \tilde{b}$. The strict inequalities hold because otherwise at least one of the instances would have the box with both minimum dimensions. As both instances belong to the same class, $E(X_1,\tilde{a},b_1) = E(X_2,a_2,\tilde{b})$ and $E(Y_1,\tilde{a},b_1) = E(Y_2,a_2,\tilde{b})$. We show that the MSI can be identified from these two instances.

As Dowsland [1987a] shows, a scaled instance of PLP remains in the same equivalence class. After scaling, the dimensions of the pallet and box may no longer be integers. Normalizing the width of the box to 1 in the above instances, we obtain instances $(X_1/b_1, Y_1/b_1, \tilde{a}/b_1, 1)$ and $(X_2/\tilde{b}, Y_2/\tilde{b}, a_2/\tilde{b}, 1)$. Because $b_1 > \tilde{b}$ and $\tilde{b} > 0$, then $1/\tilde{b} > 1/b_1$, and this result together with $a_2 > \tilde{a}$ give us $a_2/\tilde{b} > \tilde{a}/\tilde{b} > \tilde{a}/b_1$. Because an equivalence class is a convex set [Nelissen 1993], there is an instance, possibly fractional, $(X_0, Y_0, \tilde{a}/\tilde{b}, 1)$ in the class. If we multiply the dimensions by $\tilde{b}$ we obtain the instance $(X_0 * \tilde{b}, Y_0 * \tilde{b}, \tilde{a}, \tilde{b})$. We can apply the perfect partition equivalent function, obtaining $\tilde{X} = G(X_1, \tilde{a}, \tilde{b})$ and $\tilde{Y} = G(Y, \tilde{a}, \tilde{b})$. The instance $(\tilde{X}, \tilde{Y}, \tilde{a}, \tilde{b})$ satisfies the requirements to be the MSI of the class. Therefore, the class has an MSI. **Q.E.D.**

As shown above, every class has a unique MSI. And in a procedure to identify the MSI of a class, it makes no difference which box dimension we choose to minimize first.

2. Identifying the MSI

The problem of identifying the MSI could be formulated as a IP, but as in the minimization of area ratio, discussed in Chapter II, we can relax the integrality constraints and consider only normalized instances in which $b = 1$. Because the dimensions of the pallet are functions of the dimensions of the box, and the width is fixed, the problem reduces to a one-dimensional problem, where we minimize the box length. Constraints (II.7) to (II.12) are used again. But, unlike the area ratio case, the objective function is linear, and the optimal solution is attained at an extreme point. Because all coefficients in the problem are integer, the objective is rational.

Once the optimal solution is obtained, if it is fractional, say $(\hat{X}, \hat{Y}, \hat{a}, 1)$, with $\hat{a} = c/d$, where the greatest common divisor of c and d equals 1, we can scale it up and obtain an integral solution, $(\hat{X} * d, \hat{Y} * d, c, d)$, with minimum integer dimensions.

If we model the problem as an LP, constraints (II.8), (II.9), (II.11) and (II.12) can be converted to greater than or equal to inequalities if we observe that, for integer-valued variables, the minimum surplus must be 1 because all coefficients are integers.

Therefore, constraints (II.7) to (II.12) can be replaced by

$$X - i*a - j*b \geq 0, \qquad \forall (i,j) \in E(\hat{X},\hat{a},\hat{b}), \tag{III.1}$$

$$i*a + (j+1)*b - X \geq 1, \qquad \forall (i,j) \in E(\hat{X},\hat{a},\hat{b}), \tag{III.2}$$

$$(\lfloor \hat{X}/\hat{a} \rfloor + 1)*a - X \geq 1, \tag{III.3}$$

$$Y - f*a - g*b \geq 0, \qquad \forall (f,g) \in E(\hat{Y},\hat{a},\hat{b}), \tag{III.4}$$

$$f*a + (g+1)*b - Y \geq 1, \qquad \forall (f,g) \in E(\hat{Y},\hat{a},\hat{b}), \tag{III.5}$$

$$(\lfloor \hat{Y}/\hat{a} \rfloor + 1)*a - Y \geq 1. \tag{III.6}$$

In this LP, our objective is to minimize a, and we can observe that at least one of constraints (III.2), (III.3), (III.5), and (III.6) will be binding in the optimal solution, or we could reduce a even further. We call these *efficiency* constraints. Constraints (III.1) and (III.4) guarantee the feasibility of the partitions, so we call them *fitting* constraints.

If we are given an integer instance (X',Y',a',b'), we can determine if it is the MSI by testing its dimensions against constraints (III.1) to (III.6). If there are fitting constraints binding for both X and Y, and at least one of the efficiency constraints is binding, this means that we cannot reduce the values of a and b. In this case, instance (X',Y',a',b') is the MSI. If none of the efficiency constraints is binding, then we could reduce the dimensions, and (X',Y',a',b') is not the MSI.

As seen above, the identification of the MSI is a one-dimensional problem, up to a scalar multiplication, and an integer solution has at least one efficiency constraint binding.

3. Bounds on the Dimensions of the MSI of an Equivalence Class

With upper bounds on box and pallet dimensions of the MSI of a class (our class representative), we find all equivalence classes with at most a selected number of boxes. For instance $(\hat{X},\hat{Y},\hat{a},\hat{b})$, let $\hat{A}_x = \lfloor \hat{X}/\hat{a} \rfloor$ and $\hat{A}_y = \lfloor \hat{Y}/\hat{a} \rfloor$. Dowsland [1987a] proposes bounds for a and b, $a \leq B_x + 1$ and $b \leq A_x + 1$, when computing all equivalence classes with at most a selected number of boxes. Unfortunately, these bounds are incorrect, as can be verified with instance (104, 90, 15, 13), MSI of its class. For this instance, $A_x = 6$, $A_y = 6$, $B_x = 8$, $B_y = 6$, with $a = B_x + B_y + 1$ and $b = A_x + A_y + 1$. From a rearranged

version of the LP used to determine the MSI, we establish below that $a \le B_x + B_y + 1$ and $b \le A_x + A_y + 1$.

The LP (Primal) is given as follows:

Indices:

i efficient partitions on length, $i = 0,..., \hat{A}_x$.

f efficient partitions on width, $f = 0,..., \hat{A}_y$.

Data:

px_i number of V-boxes in partition i of the length.

py_f number of H-boxes in partition f of the width.

Variables:

X, Y, a, b as previously defined.

Formulation:

$Min\ b$

subject to

$$X - i * a - px_i * b \ge 0, \qquad \forall i \in \{0,1,\ldots,\hat{A}_x\}, \tag{III.7}$$

$$i * a + (px_i + 1) * b - X \ge 1, \qquad \forall i \in \{0,1,\ldots,\hat{A}_x\}, \tag{III.8}$$

$$(\hat{A}_x + 1) * a - X \ge 1, \tag{III.9}$$

$$Y - f * a - py_f * b \ge 0, \qquad \forall f \in \{0,1,\ldots,\hat{A}_y\}, \tag{III.10}$$

$$f * a + (py_f + 1) * b - Y \ge 1, \qquad \forall f \in \{0,1,\ldots,\hat{A}_y\}, \tag{III.11}$$

$$(\hat{A}_y + 1) * a - Y \ge 1. \tag{III.12}$$

In this case, we are minimizing b. Because the problem can be reduced to an one-dimensional problem, it makes no difference if we minimize a or b, because the result is the same.

An inspection of constraints (III.7) and (III.8) shows that the addition of the constraints corresponding to the same value of i in each set bounds b below by 1, i.e.,

$(X - i*a - px_i*b) + (i*a + (px_i + 1)*b - X) = b \geq 1$. Therefore, when (Primal) is feasible, it has an optimal solution, with objective function value greater than or equal to 1. In this optimal solution, at least four constraints are binding because it is a four-dimensional LP and the variables have no nonnegativity constraints [Bertsimas and Tsitsiklis 1991].

Because the MSI of a class has one perfect partition in each dimension, then, at least two fitting constraints, one in the length and one in the width, are tight in the optimal solution. Also, at least one efficiency constraint is binding.

The dual of (Primal) has only four rows, and its bases are 4×4 matrices. This characteristic makes it easer to work with the dual when computing the bound on b.

Let $\mathbf{s} = (s_{1,0},...,s_{1,A_x},s_{2,0},...,s_{2,A_x},s_3,s_{4,0},...,s_{4,A_y},s_{5,0},...,s_{5,A_y},s_6)$, be the vector of dual variables of the (Primal). The dual (Dual) of (Primal) is given by

$$Max \quad \sum_{i=0}^{A_x} s_{2i} + s_3 + \sum_{f=0}^{A_y} s_{5f} + s_6$$

subject to

$$\sum_{i=0}^{A_x} (s_{2i} - s_{1i}) * i + (A_x + 1) * s_3 + \sum_{f=0}^{A_y} (s_{5f} - s_{4f}) * f + (A_y + 1) * s_6 = 0, \tag{III.13}$$

$$\sum_{i=0}^{A_x} ((px_i + 1) * s_{2i} - px_i * s_{1i}) + \sum_{f=0}^{A_y} ((py_f + 1) * s_{5f} - py_f * s_{4f}) = 1, \tag{III.14}$$

$$\sum_{i=0}^{A_x} (s_{1i} - s_{2i}) - s_3 = 0, \tag{III.15}$$

$$\sum_{f=0}^{A_y} (s_{4f} - s_{5f}) - s_6 = 0, \tag{III.16}$$

$\mathbf{s} \geq \mathbf{0}$.

Let $\boldsymbol{P}$ be the matrix of technological coefficients of the (Primal). Then $\boldsymbol{P^T}$ has the following structure:

$$\begin{bmatrix} 0 & \dots & -A_x & 0 & \dots & A_x & A_x+1 & 0 & \dots & -A_y & 0 & \dots & A_y & A_y+1 \\ -px_0 & \dots & -px_{A_x} & px_0+1 & \dots & px_{A_x}+1 & 0 & -py_0 & \dots & -py_{A_y} & py_0 & \dots & py_{A_y}+1 & 0 \\ 1 & \dots & 1 & -1 & \dots & -1 & -1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 & 0 & 1 & \dots & 1 & -1 & \dots & -1 & -1 \end{bmatrix}$$

In every optimal dual solution, there is an optimal basis that contains one column corresponding to a perfect *X*-partition, and another column corresponding to a perfect *Y*-partition. This follows from the (Primal), in which there is always one binding X-partition row and one binding Y-partition row.

Therefore, two of the columns in the basis look like this:

$$\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Also, the optimal basis contains at least one column corresponding to a binding efficiency constraint in the primal, or otherwise the dual objective function would have value zero. Therefore, the basis contains at least one of the following columns:

$$\begin{bmatrix} \cdot \\ \cdot \\ -1 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} \cdot \\ \cdot \\ 0 \\ -1 \end{bmatrix}.$$

Considering the conditions above, the optimal basis has one of the following layouts, up to an exchange in the last two rows:

$$1)\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \; 2)\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & 0 & -1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \; 3)\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \text{ or } \; 4)\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}.$$

Let $\boldsymbol{B}$ be a basis, and let $\mathbf{s}_b = (s_p, s_q, s_r, s_s)$ be the corresponding vector of basic variables. The coefficients of the first two rows of the basis are identified as $b_{k,l}, k = 1,2, l = 1,2,3,4$, where k indicates the row and l the column. If the column corresponds to a fitting constraint in the (Primal), its coefficients are identified by $\overline{b_{k,l}}$.

The cost coefficients for the (Dual) objective function take value 0 for fitting constraints, and 1 for efficiency constraints. Therefore, the vectors of cost coefficients corresponding to the cases above are

1) (0, 0, 1, 1),

2) (0, 0, 1, 1),

3) (0, 0, 1, 1), and

4) (0, 0, 1, 1).

In all four cases, we can reduce the system of equations to a 2×2 system.

Case 1) The system is given by

$$\overline{b_{1,1}} * s_p + \overline{b_{1,2}} * s_q + b_{1,3} * s_r + b_{1,4} * s_s = 0 \tag{III.17}$$

$$\overline{b_{2,1}} * s_p + \overline{b_{2,2}} * s_q + b_{2,3} * s_r + b_{2,4} * s_s = 1 \tag{III.18}$$

$$s_p - s_r - s_s = 0 \tag{III.19}$$

$$s_q = 0. \tag{III.20}$$

We use $s_q = 0$ and $s_p = s_r + s_s$, and obtain the following system:

$$(b_{1,3} + \overline{b_{1,1}}) * s_r + (b_{1,4} + \overline{b_{1,1}}) * s_s = 0 \tag{III.21}$$

$$(b_{2,3} + \overline{b_{2,1}}) * s_r + (b_{2,4} + \overline{b_{2,1}}) * s_s = 1. \tag{III.22}$$

Case 2) The system is given by

$$\overline{b_{1,1}} * s_p + \overline{b_{1,2}} * s_q + b_{1,3} * s_r + \overline{b_{1,4}} * s_s = 0 \tag{III.23}$$

$$\overline{b_{2,1}} * s_p + \overline{b_{2,2}} * s_q + b_{2,3} * s_r + \overline{b_{2,4}} * s_s = 1 \tag{III.24}$$

$$s_p - s_r + s_s = 0 \tag{III.25}$$

$$s_q = 0. \tag{III.26}$$

We use $s_q = 0$ and $s_r = s_p + s_s$, and obtain the following system:

$$(b_{1,3} + \overline{b_{1,1}}) * s_p + (b_{1,3} + \overline{b_{1,4}}) * s_s = 0 \tag{III.27}$$

$$(b_{2,3} + \overline{b_{2,1}}) * s_p + (b_{2,3} + \overline{b_{2,4}}) * s_s = 1. \tag{III.28}$$

Case 3) The system is given by

$$\overline{b_{1,1}} * s_p + \overline{b_{1,2}} * s_q + b_{1,3} * s_r + \overline{b_{1,4}} * s_s = 0 \tag{III.29}$$

$$\overline{b_{2,1}} * s_p + \overline{b_{2,2}} * s_q + b_{2,3} * s_r + \overline{b_{2,4}} * s_s = 1 \tag{III.30}$$

$$s_p - s_r = 0 \tag{III.31}$$

$$s_q + s_s = 0. \tag{III.32}$$

We use $s_p = s_r$ and $s_q = -s_s$, and obtain the following system:

$$(b_{1,3} + \overline{b_{1,1}}) * s_r + (\overline{b_{1,4}} - \overline{b_{1,2}}) * s_s = 0 \tag{III.33}$$

$$(b_{2,3} + \overline{b_{2,1}}) * s_r + (\overline{b_{2,4}} - \overline{b_{2,2}}) * s_s = 1. \tag{III.34}$$

Case 4) The system is given by

$$\overline{b_{1,1}} * s_p + \overline{b_{1,2}} * s_q + b_{1,3} * s_r + b_{1,4} * s_s = 0 \tag{III.35}$$

$$\overline{b_{2,1}} * s_p + \overline{b_{2,2}} * s_q + b_{2,3} * s_r + b_{2,4} * s_s = 1 \tag{III.36}$$

$$s_p - s_r = 0 \tag{III.37}$$

$$s_q - s_s = 0. \tag{III.38}$$

We use $s_p = s_r$ and $s_q = s_s$, and obtain the following system:

$$(b_{1,3} + \overline{b_{1,1}}) * s_r + (b_{1,4} + \overline{b_{1,2}}) * s_s = 0 \tag{III.39}$$

$$(b_{2,3} + \overline{b_{2,1}}) * s_r + (b_{2,4} + \overline{b_{2,2}}) * s_s = 1. \tag{III.40}$$

In each of the four cases, let *d* be the determinant of the 2×2 matrix. The matrix is a basis and all elements are integers, so $|d| \geq 1$. Therefore, the respective optimal solutions of the (Dual) are

1) $s_r + s_s = (-(\overline{b_{1,1}} + b_{1,4}) + (\overline{b_{1,1}} + b_{1,3})) \big/ d \leq |b_{1,3} - b_{1,4}| \leq A_x + 1$,

2) $s_r = (\overline{b_{1,1}} - \overline{b_{1,4}}) \big/ d \leq |\overline{b_{1,1}} - \overline{b_{1,4}}| \leq A_x$,

3) $s_r = (\overline{b_{1,2}} - \overline{b_{1,4}}) \big/ d \leq |\overline{b_{1,2}} - \overline{b_{1,4}}| \leq A_x$, and

4) $s_r + s_s = (-(b_{1,4} + \overline{b_{1,2}}) + (b_{1,3} + \overline{b_{1,1}})) \big/ d \leq |(b_{1,3} + \overline{b_{1,1}}) - (b_{1,4} + \overline{b_{1,2}})| \leq A_x + A_y + 1$.

In case 1), both coefficients are nonnegative, corresponding to efficiency constraints, and one can take value 0. Therefore, the maximum dual objective function value that can be attained is $A_x + 1$.

In cases 2) and 3), all coefficients are nonpositive, corresponding to fitting constraints, and one coefficient, in each case, can take value 0. Therefore, the maximum value that can be attained is A_x.

In case 4), if we consider the order of the rows as presented in (III.35) to (III.38), the result follows because $(b_{1,3} + \overline{b_{1,1}})$ is computed over constraints in *X*, taking maximum value $A_x + 1$, and $(b_{1,4} + \overline{b_{1,2}})$ is computed over constraints in *Y*, taking minimum value $-A_y$. If the order of the rows is exchanged, the values obtained are also exchanged, with $(b_{1,3} + \overline{b_{1,1}})$ taking maximum value $A_y + 1$ and $(b_{1,4} + \overline{b_{1,2}})$ taking minimum value $-A_x$.

Considering all four cases, we verify that $A_x + A_y + 1$ is an upper bound on all optimal solutions of the dual, and an upper bound on the optimal objective function value in the (Primal). Because the primal objective is minimizing b in the equivalence class, then the value of b in the MSI for all equivalence classes, satisfies:

$$b \le A_x + A_y + 1 \qquad \text{(III.41)}$$

If we select to minimize a in the (Primal), then the derived upper bound for a is $B_x + B_y + 1$. We obtain the bounds for *X* and *Y* by combining the bounds computed for a and b and the fitting constraints in the (Primal).

Because the bounds used by Dowsland [1987a] are incorrect, we might observe some differences when analyzing the solutions of the classes defined by restrictions (II.27) to (II.29).

4. A Simple Algorithm to Identify the MSI

Previously we presented an LP formulation for the problem of identifying the MSI of a class. This model helped us develop bounds on the sizes of box and pallet in an equivalence class. These bounds enable an even simpler method for identifying the MSI.

Given an instance of PLP, $(\hat{X}, \hat{Y}, \hat{a}, \hat{b})$ or the corresponding set of efficient partitions, we can test values for $\tilde{b}$, starting at 1, and compute the other variables, until the MSI is found. The algorithm operates with two main loops. The outer loop selects values for $\tilde{b}$, from 1 to $\min\{\hat{b}, \hat{A}_x + \hat{A}_y + 1\}$. For each value of $\tilde{b}$, we compute the range of $\tilde{a}$, $R_{\tilde{a}}$,

given by $R_{\tilde{a}} = \{a \in Z^+ : \left\lfloor \hat{B}_x * \tilde{b} / (\hat{A}_x + 1) \right\rfloor < a \le \left\lceil ((\hat{B}_x + 1) * \tilde{b} - 1) / \hat{A}_x \right\rceil\}$. The second loop enumerates values for $\tilde{a}$ in $R_{\tilde{a}}$. With $\tilde{a}$ and $\tilde{b}$, we compute $\tilde{X} = G(\hat{X}, \tilde{a}, \tilde{b})$. If the selected values for $\tilde{X}$, $\tilde{a}$, and $\tilde{b}$ satisfy the efficiency constraints, repeat the same computations for $\tilde{Y}$, or otherwise try the next value for a. If the inequalities for $\tilde{Y}$ are not satisfied, continue with the procedure. If satisfied, then instance (*X*, *Y*, *a*, *b*) is the MSI. This algorithm is simple to implement, with complexity $O(B_x^4)$, and does not require setting up an LP.

B. GENERATING EQUIVALENCE CLASSES

Dowsland [1987b] works with a subset of approximately 8,000 equivalence classes with MAR bound of up to 50 boxes. Scheithauer and Terno [1996] work with a randomly generated subset of approximately 50,000 equivalence classes with MAR bound up to 100 boxes. Instead of generating random instances, or working with a subset of the problems with a given size, we enumerate all equivalence classes with up to 100 boxes per pallet, as defined by equation (II.31). Also we use the MSI to uniquely identify each class and, therefore, record only one instance per class.

If *N* is the maximum number of boxes that can be packed on a pallet, we have

$$A_x + A_y \le N + 1, \text{ and} \tag{III.42}$$

$$B_x + B_y \le 2N. \tag{III.43}$$

We recall some definitions to demonstrate these bounds. B_x (B_y) is the maximum number of items that can be placed side by side across the length (width) of the pallet. Therefore, $B_y \le B_x \le N$ and $B_y + B_x \le 2N$. Also, any optimal must have at least A_x (A_y) items placed side by side across the length (width) of the pallet so $A_x * A_y \le N$. If $A_y = 0$, $A_x \le N$ and $A_x + A_y \le N$. If $A_y \ge 1$, $A_x \le N / A_y$ and $A_x + A_y \le N / A_y + A_y \le N + 1$.

Our *PLP Equivalence Class Generation Algorithm* (PLP-ECGA) has six main loops, and uses an ordered list to maintain the distinct generated classes for given values of a and b. The outermost loop determines the values for *b*, and goes from 1 to *N*+2,

because $A_x + A_y \leq N + 1$ (Equation III.36) and $b \leq A_x + A_y + 1$ (Equation III.35). The second loop select values for *a*, from *b*+1 to 2*N*, except when *b* = 1, when *a* can also be equal to 1. If the greatest common divisor (GCD) of a and b is greater than 1, then the second loop proceeds to the next value for *a*. Otherwise, the ordered list, with all generated classes, is emptied. The third and forth loops select among all the possible perfect partitions of the width, and define the value for *Y*. The fifth and sixth loops select among the perfect partitions of the length, and define *X*. If instance (*X*, *Y*, *a*, *b*) is the MSI; has an area ratio bound within the limit; and has not been generated before, as verified with the ordered list, then it is included in the list and is recorded. The list is necessary because different combinations of a and *b* can yield the same values for *X* and *Y*.

It is possible to verify through the algorithm that the number of equivalence classes is bounded by a polynomial in *N*, albeit a large polynomial. There are $O(N^2)$ ways of assigning values to a and *b*, corresponding to the number of pairs of relatively prime numbers less than or equal to 2*N*. More precisely, the number is given by $\sum_{k=1}^{2N} \phi(k)$, where $\phi(k)$ is the Euler phi-function, which gives the number of integers less than *k* that are relatively prime to *k* [Gallian 1998]. The loops corresponding to the width are executed $O(N^2)$ times for each pair a and *b*. The same happens with the loops corresponding to the length. Therefore, the number of equivalence classes, with area ratio bound up to *N*, is bounded above by a sixth-degree polynomial in *N*. But because the algorithm generates many instances from the same class, and the same instance more than once, empirical results suggests that the number of equivalence classes might increase with N^4.

The instances generated with the PLP-ECGA procedure are divided in groups of up to 10, 20, 50 and 100 boxes per pallet, as defined by the AR bound. Table III.1 presents in the second column the number of equivalence classes of PLP in each group. The third column contains the time required, in seconds, to generate the equivalence classes within each group on a Pentium III 600 MHz personal computer. The dimensions of the pallet and box in each MSI are stored for future reference, including the work in the next chapter.

Maximum Number of Boxes in each Group	Number of Classes in each Group	Run Time to Generate all Classes (Sec)
10	662	0
20	7,309	3
50	216,095	295
100	3,080,730	14,667

Table III.1 *Number of equivalence classes in each group.*
The first column represents the maximum number of boxes in each group; the second column represents the number of equivalence classes in each group; and the third column contains the time, in seconds, required to generate all classes in each group.

Table III.2 presents the distribution of classes in each group, where the MSI is defined with b smaller than or equal to 1, 2, 5, 10, 20, and 50.

Number of Boxes	Number of Classes	$b = 1$	$b \leq 2$	$b \leq 5$	$b \leq 10$	$b \leq 20$	$b \leq 50$
10	662	92	276	609	662	662	662
20	7,309	520	1,760	4,873	6,659	7,309	7,309
50	216,095	6,362	23,270	71,686	119,298	182,870	216,095
100	3,080,730	46,300	174,177	544,004	964,673	1,710,574	2,822,767

Table III.2 *Distribution of values of b in the MSI in each class.*
The first column defines the maximum number of items that can be packed in an instance in the group of classes covered, as given by the area ratio bound. The second column lists the number of distinct classes in each group. The following columns present the number of classes where the value of b in the MSI is less than or equal to 1, 2, 5, 10, 20, and 50.

When considering instances of PLP with at most 100 boxes, in 91% of the equivalence classes the value of b in the MSI is less than or equal to 50.

In the next chapter, we develop new bounds and new algorithms for PLP, solving all instances generated with the PLP-ECGA procedure.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SOLVING PLP

In this chapter we present a new exact algorithm, new bounding procedures, and propose a set of new heuristics. One of the new heuristics, the G5-heuristic, optimally solves all PLP instances with at most 50 boxes in the packing pattern, and generates solutions that differ from the optimal solution by at most one box for all problems with 100 or fewer boxes packed.

A. A NEW SIMPLE HEURISTIC - THE HOLLOW BLOCK HEURISTIC

As described in the previous Chapter, both the diagonal block and the angle heuristics are based on placing blocks of boxes, with a given orientation, along the diagonal of the pallet, surrounded by boxes with the opposite orientation. The diagonal heuristic restricts each diagonal block to have a width of only one box; the angle heuristic is recursive in nature, and has longer run time, when compared to the diagonal heuristic.

We propose an alternative, the *Hollow Block* (HB) heuristic to solve PLP using diagonal block patterns. This new heuristic has more flexibility than the diagonal block heuristic because it allows the diagonal elements to have more than one box across the width, and achieves shorter run times than the angle heuristic because it only looks for patterns generated by perfect partitions of the length and width. This heuristic is specifically designed to be used within a more complex heuristic, where the complete arrangement of boxes in the pallet combines many blocks with different patterns.

The heuristic works by selecting compatible pairs of perfect partitions for both length and width, and uses these partitions to define the dimensions of the elements of the hollow block. Let $(i, j) \in P(X, a, b)$, for $i > 0$ and $j > 0$, and $(f, g) \in P(Y, a, b)$, for $f > 0$ and $g > 0$, be the selected perfect partitions. If $i * a \le j * b$ and $g * b \le f * a$, the diagonal elements are formed by H-boxes, and the main elements formed of V-boxes. If $j * b \le i * a$ and $f * a \le g * b$, the diagonal elements are formed by V-boxes, and the main elements formed of H-boxes. Otherwise, there can be only four blocks, two with H-boxes, and two with V-boxes.

After defining the composition of the diagonal elements, the heuristic determines whether it is feasible to create main elements with the necessary dimensions to generate the hollow block.

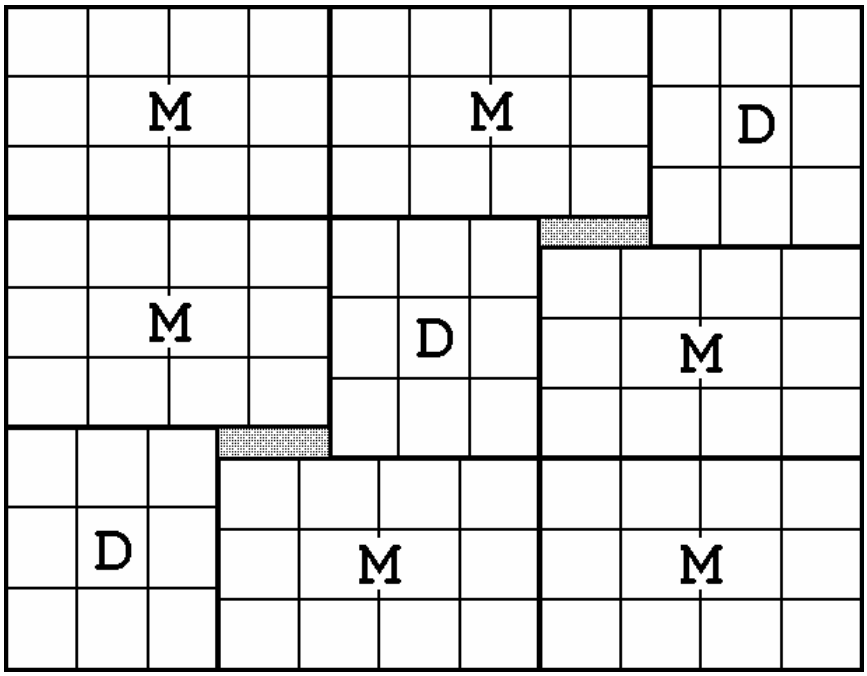

Figure IV.1 *Hollow block pattern for the instance (85, 66, 8, 7).* .

As seen in Figure IV.1, if there are d diagonal elements in a pattern, then there are $d*(d-1)$ main elements in the pattern. This happens because if we order the diagonal element from left to right, with 1 being the leftmost block and d the rightmost, each diagonal block i has $d-i$ main elements to the right, and $d-i$ above, or $\sum_{i=1}^{d} 2*(d-i) = d*(d-1)$ main elements. If the diagonal element is formed by V-boxes, as in Figure IV.1, it is feasible to create the main elements if $i \equiv 0 \bmod (d-1)$ and $g \equiv 0 \bmod (d-1)$. At this point of the heuristic, the value of d is still unknown, and the heuristic tries values that divide $GCD(i, g)$, the greatest common divisor of i and g. If the diagonal element is formed by H-boxes, the same considerations are valid, but using j and f instead. In the example in Figure IV.1, $i = 8$, $g = 6$ and $d = 3$.

The pseudocode for the HB heuristic is:

```
Best_Solution ← 0
For each (i, j) ∈ P(X, a, b), i > 0, j > 0
  For each (f, g) ∈ P(Y, a, b), f > 0, g > 0
    If i*a ≤ j*b and g*b ≤ f*a, diagonal block is H-block
      For d from 2 to GCD(j, f) + 1
        S ← d*(i*g + j*f/(d − 1))
        If S > Best_Solution then update Best_Solution
    If j*b ≤ i*a and f*a ≤ g*b, diagonal block is V-block
      For d from 2 to GCD(i, g) + 1
        S ← d*(f*j + i*g/(d − 1))
        If S > Best_Solution then update Best_Solution
    Otherwise, the solution has only four blocks
```

This heuristic has complexity similar to the two-block heuristic, because it also searches for the best combination of partitions among the efficient partitions of length and width.

B. PERFORMANCE OF SIMPLE HEURISTICS AND BOUNDS

For each equivalence class, we determine whether an optimal solution, verified with easily computed bounds, can be found using a simple heuristic. We use the one-, two-, five-block, and HB heuristics, together with the AR bound, the MP bound, Barnes' bound, and the MAR bound. We first compute all bounds and select the best one. Then we apply the heuristics in the order of increasing complexity: one-block, two-block, hollow-block, and five-block. If the bound is not attained by a heuristic, then the next one is applied.

The results are listed in Table IV.1. The first two columns are the same as in Table III.2. The following four columns have the number of instances solved to optimality using the simple heuristic listed in the column header. The last column contains the number of instances without a proven optimal solution, and are, therefore, instances where a more complex heuristic, or a better bound, can possibly produce some improvement.

Number of Boxes	Number of Classes	One-Block	Two-Block	HB	Five-Block	Classes with open results
10	662	469 (70.8)	166 (95.9)	19 (98.8)	8 (100.0)	0 (0.0)
20	7,309	4,646 (63.6)	2,156 (93.1)	236 (96.3)	203 (99.1)	68 (0.9)
50	216,095	128,204 (59.3)	63,322 (88.6)	4,455 (90.7)	13,291 (96.8)	6,823 (3.2)
100	3,080,730	1,812,852 (58.9)	840,019(86.2)	45,405 (87.7)	225,929 (95.0)	156,527 (5.0)

Table IV.1 *Absolute (and cumulative percentage, in the order of application of heuristics) performance of simple heuristics on equivalence classes in each group.*
The first two columns are the same as in Table III.2. The next four columns contain the number, and cumulative percentage, of classes with optimal packing computed by a simple heuristic and verified with the AR bound, the MP bound, Barnes' bound, or the MAR bound. The last column contains the number of classes with the best upper bound larger than the best packing pattern generated by a simple heuristic. For example, the first row shows that among the 662 distinct equivalence classes with up to 10 boxes, the one-block solved, to optimality, instances in 469 classes (70.8%), the two-block solved an additional 166 instances, totaling 635 classes (95.9%) solved with these two heuristics. The HB and five-block heuristics solved instances in the 27 remaining classes (4.1%), with up to 10 boxes packed in the pallet.

Considering instances with up to 100 boxes per pallet, the one- and two-block heuristics combined generated optimal arrangements in 86.2% of the equivalence classes. Adding the results of the other two heuristics, HB and five-block, 95.0% of the instances are solved to optimality by simple heuristics, and the result confirmed by simple bounds. This percentage does not include the cases where a better bound could prove optimality. If we consider the problems with at most 50 boxes, 96.8% of the instances are solved to optimality by simple heuristics.

C. NEW BOUNDS FOR PLP

In this section, we introduce new bounds for PLP. As seen in Table IV.1, only 5% of problems with 100 or fewer boxes remain without a proven optimal solution after applying simple heuristics and bounds. But this still represents more than 150,000 classes. In order to better identify the instances where a heuristic achieves the optimal result, but is not confirmed by a simple bound, we analyze and propose a set of new bounds.

1. Bounds Based on the Existence of Single Perfect Partitions

These bounds are applicable when, given an instance (X', Y', a', b'), there is only one perfect partition in the width (or length) of the pallet, and packing

$N' = UN(X', Y', a', b')$ boxes produces wasted area, $EW(N', X', Y', a', b')$, smaller than the corresponding dimension Y (or X) of the pallet.

The first case considered is when this single perfect partition is composed of only V-boxes or H-boxes. We call this partition a *Homogeneous Perfect Partition*. In this case, we can reduce the width or length of the pallet, and apply one of the existing procedures to compute a new bound for this smaller instance. If necessary and possible, we repeat the reduction process.

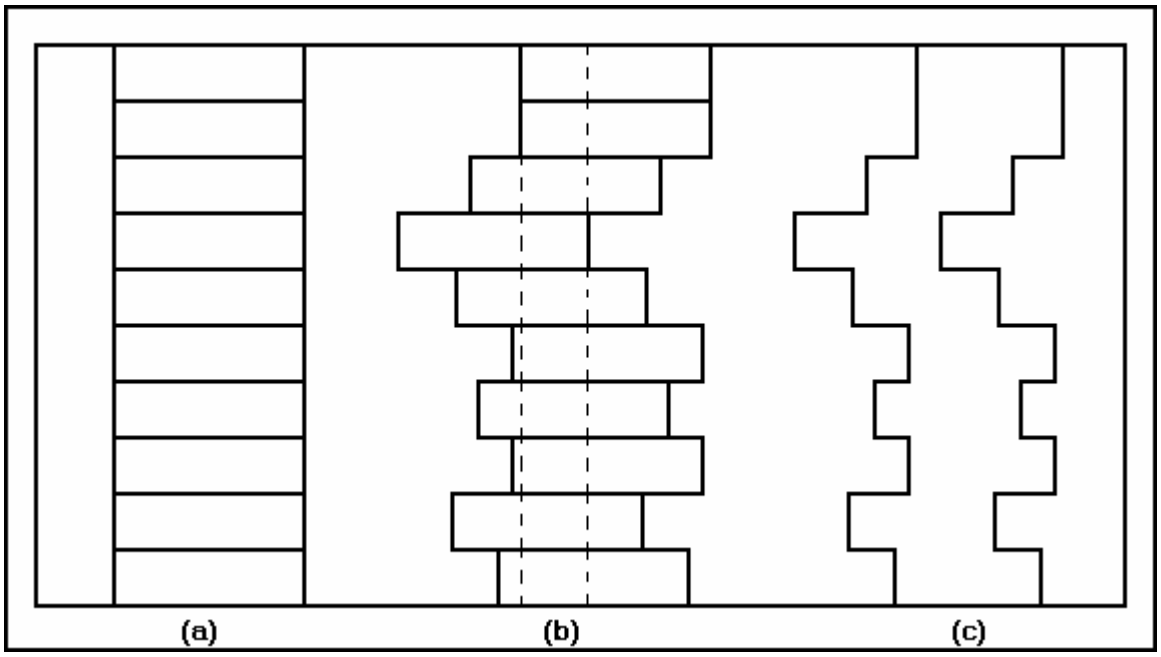

Figure IV.2 *Examples of groups of H-boxes covering a perfect partition used in the proof of Theorem IV.1. Here* $B_y = 10$ *and the dashed lines are a unit column (width 1).*

We prove the case for the width, with H-boxes.

Theorem IV.1: Let (X', Y', a', b') be an instance of PLP, with $N' = UN(X', Y', a', b')$ and $EW(N', X', Y', a', b') < X'$, and the only perfect partition in Y' is given by $B'_y * b' = Y'$. Then

$$N(X', Y', a', b') = B'_y + N(X' - a', Y', a', b')$$

Proof: The pallet can be partitioned into X columns of unit length, each of which is composed of Y unit squares, as illustrated by the dashed lines in Figure IV.2. In a normal packing, every unit square on the pallet is either completely covered or uncovered by a box. We first observe there must be at least one unit column with zero waste because

$EW(N', X', Y', a', b') < X'$. Any such unit column with zero waste must be covered with H-boxes because this corresponds to the only perfect Y-partition.

If there is a group of H-boxes completely aligned in an optimal solution, as in Figure IV.2 (a), it is easy to see that we can detach the block formed by this group, which has B_y boxes, and reconnect whatever is on either side of the block. The reduced instance has solution $N(X' - a', Y', a', b')$, and the combined solution is $B'_y + N(X' - a', Y', a', b')$.

If the group of H-boxes is present in an optimum solution in a more arbitrary shape, as in Figure IV.2 (b), then we can transform this solution to another, with the boxes forming a homogeneous block. Given an arbitrary group of B'_y H-boxes, such that the horizontal coordinates of the all boxes in this group overlap in at least one unit of length, we perform the transformation by initially removing these B'_y H-boxes from the pallet. This operation partitions the boxes in the pallet in two sets: those to the left of the removed block, the *left* set, and those to the right, the *right* set. In the next step of the transformation, all the boxes of the right set are pushed to the left, until they touch a box from the other set. Suppose that the boxes are pushed by a distance $d < a'$, and cannot be pushed any further. Then there are at least two boxes that were originally at a horizontal distance *d* from each other. Let $i \in$ left set and $j \in$ right set be these two boxes. If we divide the width in unit rows, then a portion of both *i* and *j* must be contained in at least one unit row, or otherwise they would not be touching when one is pushed to left. Figure IV.3 (a) exemplifies this situation, although the orientation of the boxes involved can be different.

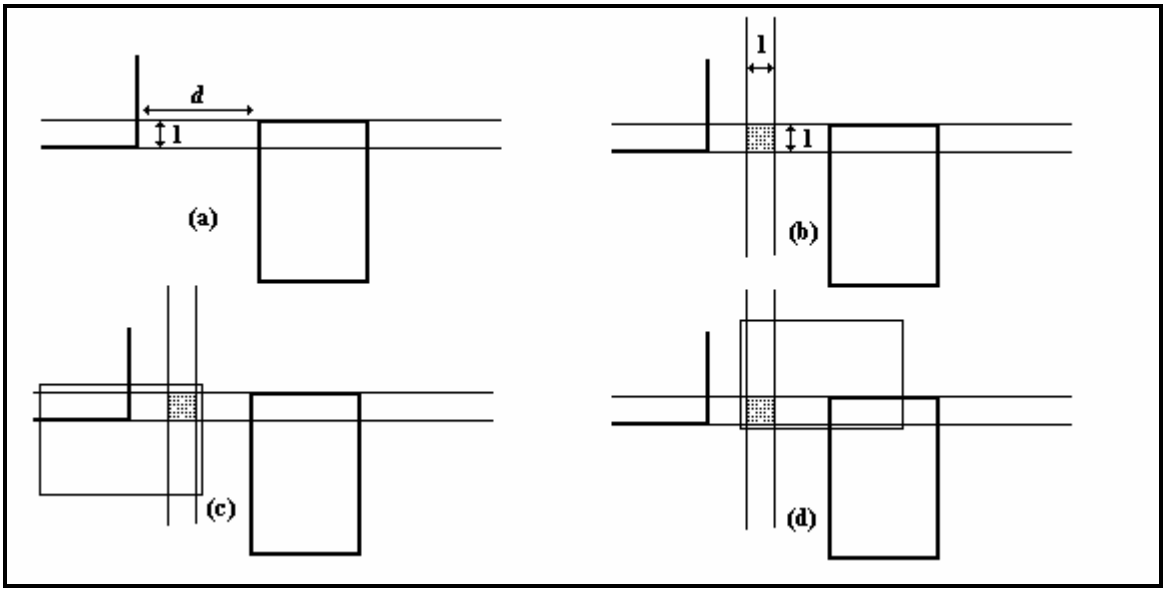

Figure IV.3 *Figure used in the proof of Theorem IV.1.*
(a) Shows the horizontal distance, d, between boxes i and j, and one unit row in which both boxes overlap; (b) indicates that at least in one unit column, the width is completely packed with H-boxes ; (c) and (b) are examples that show that an H-box cannot cover the indicated unit square, and not overlap with boxes i and j.

Between the two boxes in the original packing though, there was at least one unit column completely covered with H-boxes, as in Figure IV.3 (b). Therefore, at least one unit of the unit row segment, between i and j, is covered by an H-box. But we cannot fit an H-box in that region without overlapping with one i or j, as seen in Figure IV.3 (c) and (d). This implies that the distance between i and j is at least a', a contradiction to the initial assumption that the original distance was $d < a'$. Therefore, the movement of the boxes only stops after the right set moves a distance of at least a' units. This creates an empty region at the right edge of the pallet with length at least a', where the removed group of H-boxes can be placed, after being aligned. Figure IV.2 (c) shows the puzzle-like result of removing the group of H-boxes. **Q.E.D.**

Corollary IV.1: Let (X', Y', a', b') be an instance of PLP, with $N' = UN(X', Y', a', b')$ and $EW(N', X', Y', a', b') < X' - j * a'$. If the only perfect partition in Y is given by $B'_y * b' = Y'$, then

$$N(X', Y', a', b') = j * B'_y + N(X' - j * a', Y', a', b').$$

This result follows directly if we apply recursion to the previous result.

The cases of homogeneous perfect Y-partitions in a, and both cases of X-partitions, follow the same line of reasoning. Note that the sizes of bins to which this bound can be applied are limited to $Y \le X < a * b$, because for larger sizes if there is a homogeneous perfect partition given by $m * b$, then there is another perfect partition, $b * a + (m - a) * b$.

We also note that the procedure to rearrange boxes adopted in the proof of Theorem IV.1 only applies to homogeneous partitions that are perfect. When considering other homogeneous efficient partitions, there is at least one unit strip that is not covered with a box. In this case, the considerations associated with Figure IV.3 do not apply. In Figure IV.1, we can see an example of a packing pattern, with homogeneous efficient Y-partitions that cannot be rearranged without reducing the number of boxes packed.

We refer to this bounding procedure as the *Single Homogeneous Perfect Partition (SHPP) Bound*. Although the requirements to apply this bound may seem too restrictive to be useful, Table IV.2 shows that more than 50% of the instances with at most 100 boxes, not bounded by other elementary procedures, are bounded by this new simple bound.

Number of boxes	Number of Classes	Classes with open results from Table IV.1	One-Block	Two-Block	Hollow-Block	Five-Block	Classes with open results
20	7,309	68	20	29	0	2	17
50	216,095	6,823	1,874	1,823	0	371	2,755
100	3,080,730	156,527	37,350	35,808	0	7,726	75,643

Table IV.2 *Number of instances with an open result in Table IV.1 bounded by the SHPP Bound, in each group.*
The first three columns are from Table IV.1. The next four columns contain the number of classes with optimal packing patterns generated by the given heuristic. The last column contains the number of classes without a proven optimal solution, after applying the AR bound, the MP bound, Barnes' bound, the MAR bound and the SHPP bound.

A different bound, also based on single perfect partitions, and wasted area, can be computed, when $EW(N, X, Y, a, b) \le b$ or $EW(N, X, Y, a, b) \le a$. In this case, we use some results from Nelissen [1995].

Nelissen defines $lx_{i,j}$ as a lower bound on the number of unit rows in any optimal solution where boxes in each such unit row correspond to X-partition (i, j), and $ly_{f,g}$ as the lower bound on the number of unit columns in any optimal solution where boxes in each such unit column correspond to Y-partition (f, g). For any optimal solution, he shows if

$X - ly_{f,g} < a$, then the number of H-boxes is a multiple of *g*,

$X - ly_{f,g} < b$, then the number of V-boxes is a multiple of *f*,

$Y - lx_{i,j} < a$, then the number of V-boxes is a multiple of *j*,

$Y - lx_{i,j} < b$, then the number of H-boxes is a multiple of *i*.

We use this result in Theorem IV.2.

Theorem IV.2: Let (X', Y', a', b') be an instance of PLP, with $N' = UN(X', Y', a', b')$, $EW(N', X', Y', a', b') < b'$, and assume that the only perfect Y-partition is given by $n * a' + m * b' = Y'$. Then the number of H-boxes in the solution is a multiple of *m*, and the number of V-boxes is a multiple of *n*.

Proof: Because (*n*, *m*) is the only perfect Y-partition and $EW(N', X', Y', a', b') < b'$, there are more than $X' - b'$ unit columns where boxes in these unit columns correspond to Y-partition (*n*, *m*). Otherwise, more than b' unit columns, each with at least one unit of waste, would be present, and the total waste would be larger than $EW(N', X', Y', a', b')$. Then $ly_{n,m} > X' - b'$, $X' - ly_{n,m} < b'$, the number of H-boxes is a multiple of *m*, and the number of V-boxes is a multiple of *n*, as shown by Nelissen. **Q.E.D.**

The same results can be shown for partitions of *X*.

One example of application of this bound is the class of instance (14, 13, 4, 3). The area ratio bound is 15, with 2 units of waste. Because $EW(15,14,13,4,3) = 2 < b = 3$, and there is only one perfect *X*-partition, (2, 2), the number of H-boxes, and V-boxes, is even. The bound is reduced to 14, an even number, and the optimum solution is obtained with the two-block heuristic. We refer to the bound generated by this procedure as the *Single Perfect Partition (SPP) Bound*.

Besides the requirements on the number of H-boxes and V-boxes obtained above, another conclusion can be reached regarding the pattern necessary to pack the boxes, with a wasted area that is smaller than *b*. Because of the integrality of the sizes of boxes and pallet, there can be no wasted area at the pallet's edge, because any such region would have length, or width, at least *b*, and at least one unit for the other dimension, with an area at least *b*. Therefore, this wasted area must be located in the interior of the pallet, and the edges of the pallet are completely packed with boxes.

Number of boxes	Total Number of Classes	Classes with open results from Table IV.4	One-Block	Two-Block	Hollow-Block	Five-Block	Classes with open results
20	7,309	17	0	1	0	2	14
50	216,095	2,755	0	20	1	28	2,706
100	3,080,730	75,739	0	128	19	234	75,358

Table IV.3 *Number of instances with an open result in Table IV.2 bounded by the SPP bound in each group.*
The first three columns are from Table IV.2. The next four columns contain the number of classes with optimal packing pattern generated by the given heuristic. The last column contains the number of classes without a proven optimal solution after applying the AR bound, the MP bound, Barnes' bound, the MAR bound, the SHPP bound and the SPP bound.

2. Bounds Based on Relaxed and Restricted Classes

The concept of equivalence classes is present in several procedures developed to solve, or bound, instances of PLP. This previous work exploits the symmetry of the relation between instances belonging to the same class. If the relation between the set of feasible partitions of two instances is not symmetric, we extend the use of equivalence classes by defining relaxed and restricted classes.

Let (X, Y, a, b) and (Z, W, c, d) be instances of PLP, with set of feasible partitions given by $F(X,a,b)$, $F(Y,a,b)$, $F(Z,c,d)$, and $F(W,c,d)$. If $F(X,a,b) \subset F(Z,c,d)$ and $F(Y,a,b) \subset F(W,c,d)$, we define ($Z$, W, c, d) to be a *relaxed class* of (X, Y, a, b), and represent it $(Z,W,c,d) \succ (X,Y,a,b)$, and define ($X$, Y, a, b) to be a *restricted class* of (Z, W, c, d), $(X,Y,a,b) \prec (Z,W,c,d)$.

For example, consider the instances (69, 36, 11, 5) and (56, 29, 9, 4). The only difference between the sets of feasible partitions is that (0, 14) $\in$ F(56, 9, 4), but (0, 14) $\notin$ F(69, 11, 5). In this case, $(69,36,11,5) \prec (56,29,9,4)$. As a result, instance (56, 29, 9, 4) can be packed in more ways than instance (69, 36, 11, 5). Figure IV.4 depicts a feasible arrangement for instance (56, 29, 9, 4) in (a), and a feasible arrangement for instance (69, 36, 11, 5) in (b).

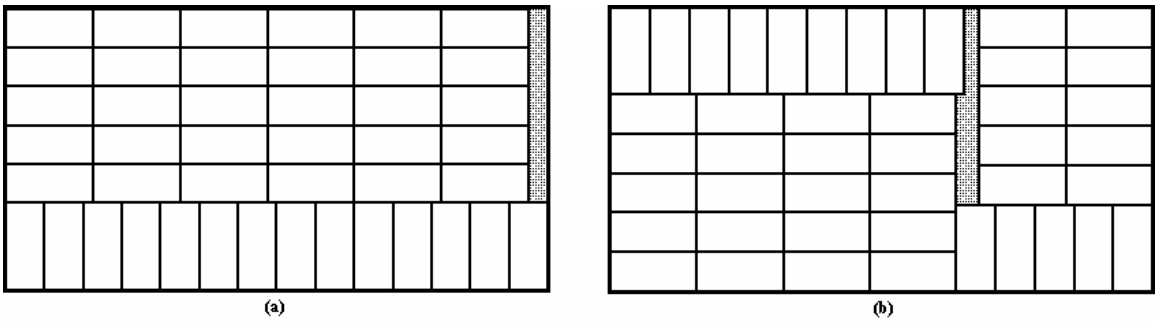

Figure IV.4 *Example of packing patterns for restricted and relaxed classes. The packing pattern in (a) is feasible for instance (56,29,9,4), but not for instance (69,36,11,5); (b) is a feasible packing pattern for both instances.*

Theorem IV.3: Given an instance (*X*, *Y*, *a*, *b*), with a feasible arrangement of the boxes on the pallet, and a second instance (*Z*, *W*, *c*, *d*), such that $(Z,W,c,d) \succ (X,Y,a,b)$, then there is a corresponding arrangement of box $c \times d$ on pallet $Z \times W$.

Proof: The proof follows directly from the proof of *Lemma* 2, in Dowsland [1984], because the proof only requires that the feasible partitions of instance (*X*, *Y*, *a*, *b*) be replicated by instance (*Z*, *W*, *c*, *d*), and this is clearly true, because $F(X,a,b) \subset F(Z,c,d)$ and $F(Y,a,b) \subset F(W,c,d)$. **Q.E.D.**

The relation between instances of relaxed and restrict classes is not symmetric, but is transitive. Therefore, if $(X,Y,a,b) \prec (Z,W,c,d)$, and $(Z,W,c,d) \prec (R,S,e,f)$, then $(X,Y,a,b) \prec (R,S,e,f)$. For example, $(69,36,11,5) \prec (56,29,9,4)$. But $(56,29,9,4) \prec (43,23,7,3)$, so $(69,36,11,5) \prec (43,22,7,3)$.

Every instance of PLP in a relaxed class is a relaxation to instances in a restricted class. As a result, an upper bound on the number of boxes that can be packed in an instance in a relaxed class is an upper bound for all its restricted classes. If we are not able to compute a tight bound for a given instance, it might be possible to compute a tighter bound on an instance in a relaxed class, and apply the bound.

Theorem IV.4: If (*X*, *Y*, *a*, *b*) and (*Z*, *W*, *c*, *d*) are instances of PLP, and if $(X,Y,a,b) \prec (Z,W,c,d)$, then $N(X, Y, a, b) \le N(Z, W, c, d)$.

Proof: Suppose it is not true, and $N(X, Y, a, b) > N(Z, W, c, d)$. Then there exist a feasible arrangement of $a \times b$ boxes on the $X \times Y$ pallet with $N(X, Y, a, b)$ boxes packed. Because (*Z*, *W*, *c*, *d*) belongs to a relaxed class of (*X*, *Y*, *a*, *b*), by Theorem IV.3, every feasible arrangement obtained in (*X*, *Y*, *a*, *b*) can be replicated in (*Z*, *W*, *c*, *d*). Therefore, the

corresponding arrangement can be produced in (Z, W, c, d), and $N(X, Y, a, b) \leq N(Z, W, c, d)$. This is a contradiction. **Q.E.D.**

As an example, $UN(26, 19, 7, 3) = 23$, when using the previously described bounds. But $(26,19,7,3) \prec (18,13,5,2)$, because $(0,9) \in F(18,5,2)$, and $UN(18, 13, 5, 2) = 22$, obtained with Barnes' bound within the class. Therefore, the upper bound for instance (26, 19, 7, 3) can be reduced to 22, and the solution obtained with the five-block heuristic is verified to be optimal by this new bound. We refer to the bound computed with this procedure as the *Relaxed Class (RC) Bound*.

Table IV.4 presents the results of this new bound when applied to the instances from Table IV.3 not yet proven optimal by other bounds. While the number of instances where this bound is useful is small, when compared with the total number of classes, we can see that this bound is tighter than the previous bounds on a large fraction of the classes still considered open.

Number of boxes	Total Number of Classes	Classes with open results from Table IV.5	One-Block	Two-Block	Hollow-Block	Five-Block	Classes with open results
20	7,309	14	0	1	0	0	13
50	216,095	2,706	115	589	2	218	1,782
100	3,080,730	75,358	4,298	16,260	61	8,107	46,632

Table IV.4 *Number of instances with an open result in Table IV.3 bounded by the RC bound in each group.*
The first three columns are from Table IV.3. The next four columns contain the number of classes with optimal packing pattern generated by the given heuristic. The last column contains the number of classes without a proven optimal solution after applying the AR bound, the MP bound, Barnes' bound, the MAR bound, the SHPP bound, the SPP bound, and the RC bound.

One way to identify relaxed classes is by applying Dowsland's [1984] minimization of area ratio technique. When the instance that minimizes the area ratio is located at the open boundary of an equivalence class, then this instance does not belong to that class, although we can get arbitrarily close to it. This instance does not belong to the same class because its set of feasible partitions is larger, with at least one extra feasible partition.

In our previous example, the instance (18, 13, 5, 2) is obtained when trying to minimize the area ratio for the class of (26, 19, 7, 3). When minimizing the area ratio bound in the relaxed class, we verify that this is attained at a non-integral interior point

$(2\sqrt{8}+4, \sqrt{8}+4, \sqrt{8}, 1)$, with a ratio of 23.31. This area ratio is not small enough to tighten the bound, but a solution obtained by scaling, and rounding to integer values, (482, 341, 141, 50), is bounded by Barnes' procedure at value 22.

Another possible result is that another open boundary solution is generated, and the procedure is repeated until the bound is tighter, or the solution is an interior point. An example of this *chain* of relaxed classes can be obtained with the instance (54, 30, 11, 5), with three relaxed classes being generated before the bound is tightened. The relation among these classes is $(54,30,11,5) \prec (44,24,9,4) \prec (34,18,7,3) \prec (24,12,9,2)$. The area ratio bound in the last class is 28, while it is 29 in the others.

As the relaxed classes can be generated based on minimizing the area ratio bound, in some cases, the last class generated in the procedure finally has zero wasted area, but does not tighten the bound. Because there is no wasted area, every feasible partition used when packing the instance in the restricted class must correspond to a perfect partition in the relaxed class, or there would be some wasted area. But if only one feasible partition, in the length or width, corresponds to a perfect partition in the relaxed class, then the solution can be obtained with the two-block heuristic, or it is not feasible. If we apply the two-block heuristic and don't obtain the expected solution, then the bound in the restricted class can be reduced by at least one unit.

This happens, for example, with instance (23, 19, 8, 3). The area ratio bound for this instance is 18. When minimizing this bound in the class, we obtain the instance (15, 12, 5, 2), in the open boundary. Because 15 is a multiple of 5 and 12 is even, we can pack it using a homogeneous block with 18 boxes and there is no wasted area. This is the only pattern that can be used to pack 18 boxes and it is not feasible for instance (23, 19, 8, 3), as it uses the X-partition (3, 0). If there are feasible packing patterns with 18 boxes for instance (23, 19, 8, 3), with a different arrangement, then this arrangement could be replicated in the relaxed class, so the bound for instance (23, 19, 8, 3) can be reduced to 17.

3. Bound Based on Similarity of Classes

The properties of the prior section can sometimes be extended in a way that a class, with a similar set of efficient partitions, can be used to calculate a bound for the solution of another class.

One such situation exists when the MSI of the class has a homogeneous perfect partition. As seen previously, if a perfect partition of this type is present in the solution, then it can be transferred to the edge of the pallet. In this case, we can solve a problem with reduced dimensions for the pallet and add this homogeneous block to generate a solution to the original problem. If this homogeneous partition is the only perfect partition, then we have a tighter bound. But when there is more than one perfect partition, then the bound cannot be used directly.

Two possible situations can happen: the optimal packing pattern uses the homogeneous partition, or not.

If the homogeneous partition is present in the solution, then there is a solution where this partition is located in the right, or top, edge of the pallet, as shown in Theorem IV.1. In this case, we can solve an instance with reduced dimension by dividing the pallet into two rectangles. One is perfectly packed with a homogeneous block, and the other is packed by some other procedure.

Otherwise, if the homogeneous partition is not used in the optimal packing pattern, then it may be possible for another class to exist in which all efficient partitions, except for this homogeneous partition, are feasible. In this case, any pattern produced in the initial class, without using this homogeneous partition, can be replicated. But if, in this related class, it is shown that such a pattern does not exist, then the same is also true in the initial class.

The larger of the bounds obtained in these two situations is a valid upper bound for the class. The instance (37, 30, 8, 3) is an example of the use of this bound. The set of perfect Y-partitions contains two partitions: (3, 2) and (0, 10). The best bound for this instance, using the methods proposed above, is 46 boxes. If the solution contains a (0, 10) partition then $N(37,30,8,3) = \max\{N(37-8*i,30,8,3)+10*i,\ i=1,2,3,4\} = 45$. If the partition (0, 10) is not used, then all other feasible partitions are also feasible to the instance (24, 19, 5, 2). But the optimal number of packed boxes for instance (24, 19, 5, 2) is 45, using the area ratio bound. Therefore, if the Y-partition (0, 10) is used, the best result is 45. If it is not used, the best result is also 45. Then $UN(37, 30, 8, 3) = 45$.

We refer to this bounding procedure as the *Combined Perfect Partition and Relaxed Class (CPPRC) Bound*. Table IV.5 presents the results of this new bound when applied to the instances from Table IV.4 not yet proven optimal by other bounds.

Number of boxes	Total Number of Classes	Classes with open results from Table IV.4	One-Block	Two-Block	Hollow-Block	Five-Block	Classes with open results
20	7,309	13	0	0	0	1	12
50	216,095	1,782	0	24	3	72	1,683
100	3,080,730	46,632	0	289	21	770	45,552

Table IV.5 *Number of instances with an open result in Table IV.5 bounded by the CPPRC bound in each group.*
The first three columns are from Table IV.5. The next four columns contain the number of classes with optimal packing pattern generated by the given heuristic. The last column contains the number of classes without a proven optimal solution after applying the AR bound, the MP bound, Barnes' bound, the MAR bound, the SHPP bound, the SPP bound, the RC bound, and the CPPRC bound.

4. Extensions to the LP Bound

Even when the identification of a relaxed class does not help directly in the computation of a tighter bound, it still can be used to help obtain better bounds using the LP approach described in Section III of Chapter II.

Suppose we are computing an upper bound on a given instance, and that this instance has a relaxed class with smaller dimensions, and less wasted area. We can use the MSI of this relaxed class when computing the LP bound, but only considering partitions that are feasible in the original instance.

Let (Z, W, c, d) be an instance of PLP. Let $(X,Y,a,b) \succ (Z,W,c,d)$. Then constraints (II-20) to (II-27) are replaced by

$$H = \sum_{(i,j) \in F(Z,c,d)} \frac{i * x_{i,j}}{b}, \tag{IV.1}$$

$$V = \sum_{(f,g) \in F(W,c,d)} \frac{f * y_{f,g}}{b}, \tag{IV.2}$$

$$\sum_{(i,j) \in F(Z,c,d)} x_{i,j} = Y, \tag{IV.3}$$

$$\sum_{(f,g) \in F(W,c,d)} y_{f,g} = X, \tag{IV.4}$$

$$\sum_{(i,j) \in F(Z,c,d)} a*i*x_{i,j} - \sum_{(f,g) \in F(W,c,d)} b*g*y_{f,g} = 0\,, \tag{IV.5}$$

$$\sum_{(i,j) \in F(Z,c,d)} b*j*x_{i,j} - \sum_{(f,g) \in F(W,c,d)} a*f*y_{f,g} = 0\,, \tag{IV.6}$$

$$x_{i,j} \geq 0, \quad \forall (i,j) \in F(Z,c,d)\,, \tag{IV.7}$$

$$y_{f,g} \geq 0, \quad \forall (f,g) \in F(W,c,d)\,. \tag{IV.8}$$

The advantage of using this new set of constraints is that the linear relaxation is tighter, and thereby may lead to an improved bound.

Some additional changes are performed in the LP bound procedure described in Section III of Chapter II. Nelissen [1995], among other extensions to the LP bound, proposes solving a knapsack problem with the wasted area of an optimal solution as the capacity of the knapsack, and the wasted area in each unit column, or row, as the weight. This idea can be inserted directly in the LP formulation by defining a *target* optimal wasted area, *Waste*. If this targeted wasted area value cannot be achieved, then the bound can be reduced, and another target waste can be defined. Two extra constraints, relative to this target waste, are included in the formulation:

$$\sum_{(i,j) \in F(Z,c,d)} (X - (i*a + j*b))*x_{i,j} = Waste\,, \tag{IV.9}$$

$$\sum_{(f,g) \in F(W,c,d)} (Y - (f*a + g*b))*y_{f,g} = Waste\,. \tag{IV.10}$$

One example of the usefulness of this improved LP bound is observed with instance (116, 74, 10, 9), with a best-known upper bound of 95, but the best-known packing contains 94 boxes. This is one of the instances reported by Scheithauer [2000] where the best-known packing does not attain the best-known upper bound. The instance (68, 42, 6, 5) is the last class generated in the computation of the relaxed classes using the minimization of area ratio bound procedure. When applying the new LP bound, the bound is reduced to 94.

5. Performance of New Bounds

Table IV.6 presents the cumulative results displayed previously from Tables IV.1 to Table IV.5. It does not contain complete results for the extended LP bound, because this new procedure was applied to a small set of the remaining *open* problems.

Number of boxes	Classes with open results from Table IV.1	Instances bounded with SHPP Bound	Instances bounded with SPP Bound	Instances bounded with RC Bound	Instances bounded with CPPRC Bound
20	68	51	3	1	1
50	6,823	4,068	49	924	99
100	156,527	80,788	381	28,661	1,083

Table IV.6 *Cumulative results of the SHPP, SPP, RC and CPPRC bounds.*
The first two columns are from Table IV.1. The values on the SHPP column come from Table IV.2. The values on the SPP column come from Table IV.3. The values on the RC column come from Table IV.4. The values on the CPPRC column come from Table IV.5. These results are obtained when applying the bounds in column order.

As shown in Table IV.6, more than 70% of the instances not bounded exactly by the AR Bound, MP Bound, Barnes' Bound and MAR Bound, are bounded using the new bounds proposed in this dissertation.

Table IV.7 presents the results of the individual application of these bounds on the instances with *open* results from Table IV.1. These numbers corresponds to instances where the mentioned bound is able to tighten the bound (reduce by at least 1 box), even when not being exact.

Number of boxes	Classes with open results from Table IV.1	Bound tighten with SHPP procedure	Bound tighten with SPP procedure	Bound tighten with RC procedure	Bound tighten with CPPRC procedure
20	68	52	3	52	56
50	6,823	4,571	52	5,032	5,161
100	156,525	97,186	463	111,800	113,065

Table IV.7 *Results of individually applying the SHPP, SPP, RC and CPPRC bounds.*
The first two columns are from Table IV.1. The next four columns present the number of equivalence classes with the bound on the maximum number of boxes packed reduced with the application of the respective procedure.

In order to compare the performance of the new proposed bounds with the results obtained by Nelissen [1995] with his extended LP bound, we apply the proposed bounds to a set of 187 instances of PLP whose correct bounds are not computed by other methods. Nelissen [1995] reported that his procedure is not able to tighten the bound for 36 instances. In our case, the best-computed bound is not exact for only six instances.

Only about 1.5% of the equivalence classes of PLP with at most 100 boxes may require a more complex algorithm to deliver the optimal solution, although in some cases it may just require a better bounding procedure to confirm the optimality of a known packing pattern.

D. NEW RECURSIVE HEURISTICS FOR PLP

In this dissertation, we propose new recursive heuristics to solve PLP. The first heuristic, named the G5-heuristic, can be regarded as a combination of the heuristics proposed by Nelissen [1993], Scheithauer and Terno [1996], and Morabito and Morales [1998]. The second one, named Higher Order Non-Guillotine (HONG) heuristic, solves some of the problems encountered when dealing with problems with more than 50 boxes to pack.

1. The G5-Heuristic

The heuristic proposed by Nelissen [1993] combines a few simple heuristics into a recursive heuristic. The number of recursion steps is limited to reduce run time.

The heuristic proposed by Scheithauer and Terno [1996], the G4-heuristic, recursively applies the four-block heuristic. There is no a priori limitation on the number of recursive calls, and bounds are used to decide if further searches in the recursion tree are necessary. The G4-heuristic does not identify optimal solutions that do not present G4-patterns, and unlimited recursion may result in long run times when solving instances with a large number of boxes.

The heuristic proposed by Morabito and Morales [1998], referred to here as M&M, uses the idea of 1^{st}-order cuts, discussed in Section I.E, and the solutions generated present 1^{st}-order patterns. This heuristic has a larger solution space than the G4-heuristic, and it may take longer to enumerate all solutions.

The G5-heuristic also looks for 1^{st}-order patterns, but tries to reduce run times by applying the hollow block heuristic. The hollow block pattern, as shown in Figure IV.1, is a G4-pattern, and also a 1^{st}-order pattern, but can be identified much faster by the hollow block heuristic. Another characteristic of the G5-heuristic is that there is only one level of recursion.

There are four main loops in the algorithm, sequentially assigning the dimensions of the four blocks in corners of the packing pattern. The fifth, or central, block has its dimensions defined by the other blocks as in the five-block heuristic. Each of these five blocks has a packing pattern computed using the hollow block, one-, two-, and five-block heuristics, or the G5-heuristic, but with no additional recursive call.

The wasted area within each block is also computed. At each step of the algorithm, after defining the dimensions of a block, the cumulative sum of wasted areas is computed, preventing the exploration of patterns producing too much waste.

In order to avoid looking at symmetric patterns, the block with the most boxes packed, with the exception of the central block, is defined to be the first block at the lower left corner. Whenever a pattern is produced with another block having a larger number of boxes, the execution of the algorithm moves to the next pattern. This is a different strategy from the G4 and M&M heuristics.

As implemented in the G4-heuristic, we keep track of all partial patterns produced. Therefore, the solution to the packing problem of a block with given dimensions is stored when it is first computed, and is used again whenever a block with the same dimensions is encountered later in the procedure.

Morabito and Morales [1998] present the computation results obtained with their heuristic to selected instances from the literature, with run times recorded on a Pentium 100 MHz personal computer. Scheithauer has an implementation of the G4-heuristic available for download from the Internet [CADAP 2002]. We run both G4 and G5 heuristics, on the same set of problems, on a Pentium 133 MHz personal computer, so the run times could be compared with those reported by Morabito and Morales (Table IV.7). As described by Morabito and Morales, problems D1 and D2 are from Dowsland [1984], N1 to N5 from Nelissen [1994], and ST1 to ST5 from Scheithauer and Terno [1996].

ID	Instance	MSI	N	Run Time (Sec)		
				M&M	G4	G5
D1	(22,16, 5, 3)	(22, 16, 5, 3)	23	0.10	0.00	0.00
D2	(86, 82, 15, 11)	(23, 22, 4, 3)	42	0.50	0.05	0.06
N1	(43, 26, 7, 3)	(43, 26, 7, 3)	52*	1.60	0.06	0.50
N2	(87, 47, 7, 6)	(87, 47, 7, 6)	97	46.30	1.21	2.42
N3	(153, 100, 24, 7)	(109, 71, 17, 5)	90	0.10	1.21	0.01
N4	(42, 39, 9, 4)	(42, 39, 9, 4)	45	2.00	0.11	0.25
N5	(124, 81, 21, 10)	(64, 41, 11, 5)	47	5.50	0.16	0.11
ST1	(40, 25, 7, 3)	(40, 25, 7, 3)	47	2.10	0.11	0.19
ST2	(52, 33, 9, 4)	(52, 33, 9, 4)	47	3.10	0.11	0.19
ST3	(57, 44, 12, 5)	(57, 44, 12, 5)	41	0.90	0.11	0.13
ST4	(56, 52, 12, 5)	(56, 52, 12, 5)	48	2.20	0.11	0.21
ST5	(300, 200, 21, 19)	(127, 85, 9, 8)	149	0.10	9.23	0.01

Table IV.8 *Run times obtained with three different heuristics, when solving a selected set of problems from the literature.*
** Indicates that the solution is not optimal.*

The G5-heuristic is slower than the G4-heuristic, and faster than the M&M heuristic in almost all instances.

Scheithauer [2000] provided a list with 206 instances, from a larger set called *Cover II* by Nelissen [1993], containing instances satisfying constraints (II.28), (II.29), and (II.31). These 206 instances are not solved to optimality by Nelissen's heuristic. Scheithauer [2000] reports that the G4-heuristic is able to solve 167 of these instances. The G5-heuristic is able to solve the same 167 instances, and solves two additional instances, (121, 120, 16, 9), and (107, 65, 10, 7). These two instances have solutions with 1st-order patterns that are not G4-patterns. On instances in which both heuristics find the optimal solution, the G4-heuristic is faster than the G5-heuristic.

Table IV.9 presents the number of instances not previously solved with the simple heuristics mentioned in Section C with a proven optimal solution obtained with the G5-heuristic.

Number of boxes	Total Number of Classes	Classes with Open Results from Table IV.6	Solved with G5-Heuristic	Classes with open results
20	7,309	12	10	2
50	216,095	1,683	1,449	234
100	3,080,730	45,552	38,546	7,006

Table IV.9 *Number of instances from table IV.6 without a previously proven optimal solution, solved using the G5-heuristic.*
The first three columns are from Table IV.6. The next column contains the number of classes with optimal packing pattern generated by the G5-heuristic. The last column contains the number of classes without a proven optimal solution after applying the AR bound, the MP bound, Barnes' bound, the MAR Bound, the SHPP bound, the SPP bound, the RC bound, and the CPPRC bound.

Another important result is that the difference between the solution generated by the G5-heuristic and the best possible solution, as defined by the available upper bounds, is at most one box for all problems. As it is shown in the next section, with the application of an exact algorithm, the G5-heuristic generates optimal solutions to all instances of PLP with AR bound up to 51 boxes, and approximately 99.999% of all instances with AR bound of up to 100 boxes, differing at by one box in the 0.001% (60 classes) remaining instances. Therefore, using a procedure exploring non 1st-order patterns can improve the solution by at most one box.

2. Higher Order Non-Guillotine Heuristics

As the number of boxes that can be packed increases, so does the variety of the optimal patterns. Morabito and Morales [1998] present a non 1st-order pattern solution to instance (43, 26, 7, 3), obtained using a MIP model of PLP and solved using GAMS 2.50 [GAMS 2000] with the OSL 2 solver [IBM 2000]. They report that the solver needed more than 30 minutes, on a Pentium 100 MHz personal computer, to find the optimal solution, shown in Figure IV.5. The borders of each block are reinforced so it is easier to observe that there are 8 blocks, and that these blocks do not form a 1st-order pattern. This is one of the instances not solved by the G5-heuristic in the COVER II set.

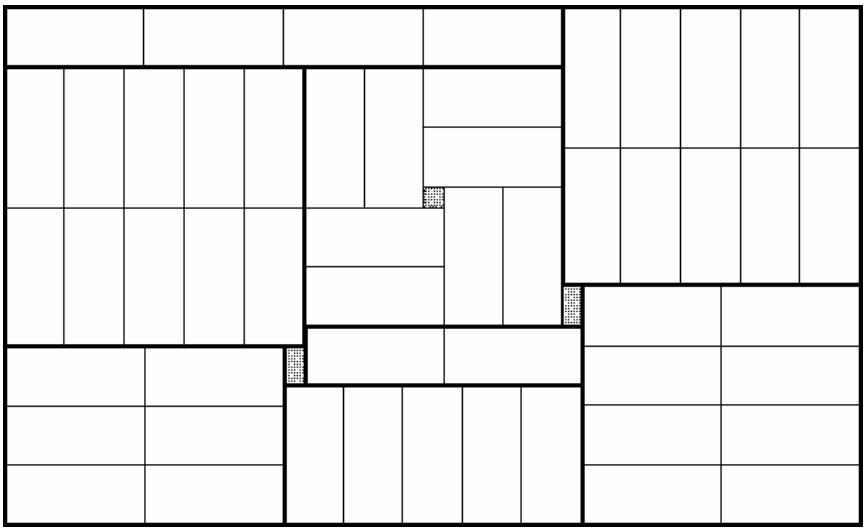

Figure IV.5 *Non 1st-order pattern solution for the class of instance (43, 26, 7, 3).*

The *Higher-Order Non-Guillotine* (*HONG*) heuristics have been developed, initially, to verify if other instances in the same set present the same type of arrangement. They work by dividing the pallet in at most eight blocks, distributed as shown in Figure IV.6. As can be seen, all three patterns are not 1st-order. The pattern in Figure IV.6 (a) is a *vertical* pattern, because it is possible to draw a vertical line crossing four blocks. A *horizontal* pattern, where a horizontal line crosses four blocks, is shown in (b). The last pattern, (c), is named a *central* pattern.

The implementation of the HONG heuristics is an extension of the G5-heuristic, but has up to five additional loops, for a total of nine nested loops. Each of these eight blocks can be packed using the non-recursive G5-heuristic. We implemented the three versions of the heuristic. We are not able to identify any characteristics of an instance indicating which

version is the best one to apply in each case, and because of the relatively (when compared with the G5-heuristic) long run times associated with each version of the algorithm, we propose to execute all versions in parallel, halting when a solution is obtained by one of the three versions.

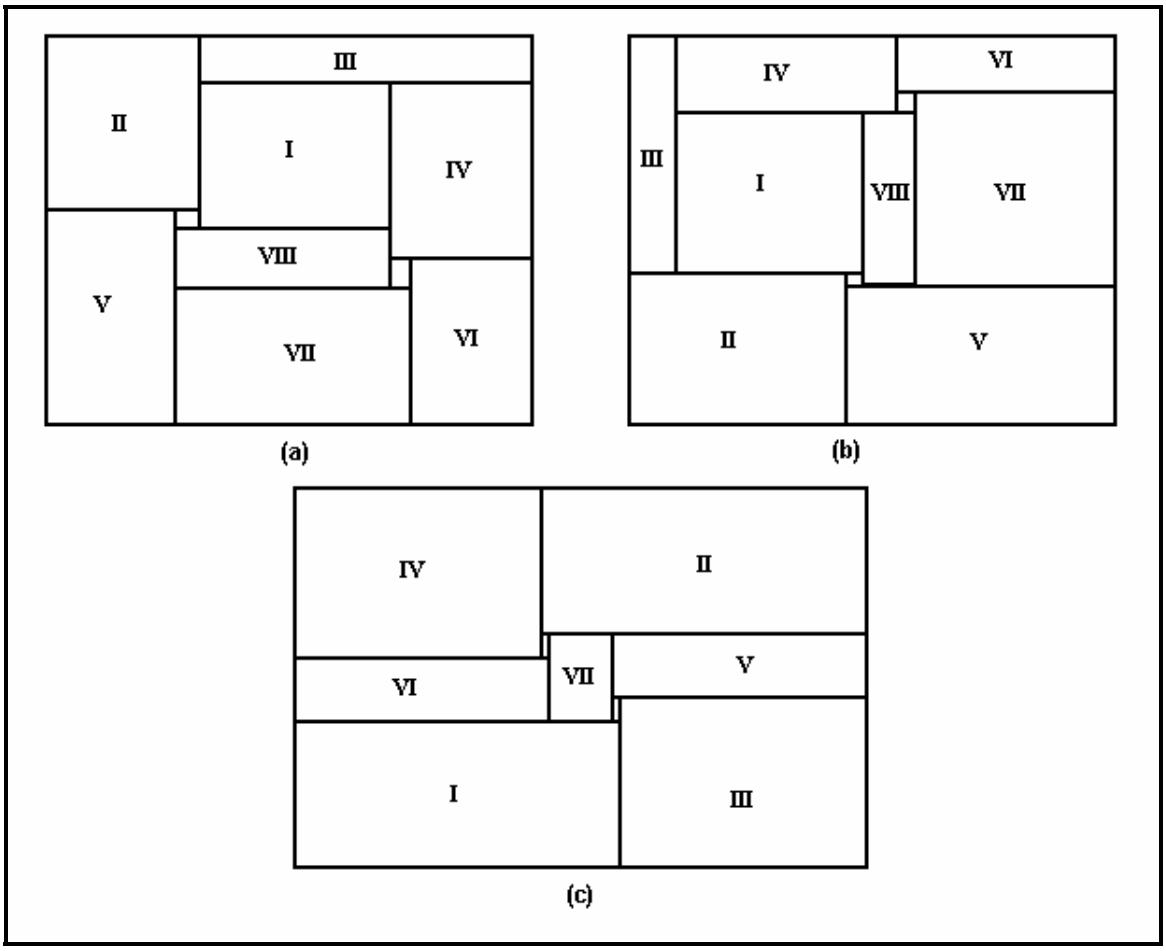

Figure IV.6 *Non 1st-order patterns explored by the HONG heuristic.*
The pattern in (a) is a vertical pattern; (b) shows a horizontal pattern; and (c) is a central pattern.

Table IV.9 lists the instances of the COVER II set, not previously solved by other heuristics, which are solved using the HONG heuristics. The run times shown correspond to the time required by a Pentium III 600 MHz computer, using the version of the heuristic (vertical or horizontal) that obtains the optimal solution for each instance. If all versions are used in parallel, we should observe computational effort roughly three times the values listed.

Instance	Number of Boxes	Run Time (Sec)
(43,26,7,3)	53	0.50
(49,28,8,3)	57	0.99
(61,35,10,3)	71	2.63
(61,38,10,3)	77	3.66
(67,37,11,3)	75	3.36
(67,40,11,3)	81	7.51
(141,119,21,8)	99	88.17
(93,46,13,4)	82	15.26
(63,44,8,5)	69	3.80
(57,34,7,4)	69	2.64
(106,59,13,5)	96	17.84
(141,71,13,8)	96	18.13
(74,73,13,5)	82	30.45
(74, 49, 11, 4)	82	0.28
(127, 121, 23, 7)	95	1.20
(106, 59, 13, 5)	96	8.68

Instance	Number of Boxes	Run Time (Sec)
(76,74,13,5)	86	345.54
(106,100,16,7)	94	11.26
(83,82,11,7)	88	3.48
(104,69,12,7)	85	2.10
(103,86,11,8)	100	15.05
(104,71,11,7)	95	20.47
(75,51,8,5)	95	6.91
(108,71,11,7)	99	10.61
(78,51,8,5)	99	4.54
(61,38,6,5)	77	0.86
(108,65,10,7)	100	2.44
(164,83,14,11)	88	3.38
(105,53,9,7)	88	2.83
(57, 34, 7, 4)	69	1.70
(122, 86, 16, 7)	93	1.97

Table IV.10 *Instances of the set COVER II solved to optimality by the HONG heuristics. Run times correspond to the time required by the version of the algorithm (vertical, horizontal, or central) that obtains the optimal solution faster (in some cases, more than one version generates an optimal packing pattern).*

In addition to instances in Table IV.9, the HONG heuristic is able to solve to optimality other instances not included in the COVER II set [Nelissen 1993]. Table IV.10 presents the results of the application of the HONG heuristic to the instances with open results after the using the G5 heuristic, from Table IV.10.

Number of boxes	Total Number of Classes	Classes with Open Results from Table IV.8	Solved by HONG Heuristic	Classes with open results
20	7,309	2	0	2
50	216,095	234	0	234
100	3,080,730	7,006	54	6,952

Table IV.11 *Number of instances, from table IV.8, without a proven optimal solution, solved using the HONG heuristics.*

E. A NEW EXACT ALGORITHM FOR PLP – THE HVZ ALGORITHM

Even with the development of new bounds and new heuristics, not all instances have a proven optimal solution given by a heuristic. In some situations it still might be desirable to find the optimal solution, even if it might require the extra run time of an exact algorithm. In order to solve, or verify optimality of, these remaining problems, we developed a new exact algorithm.

We initially explore different ways to represent a packing pattern, considering the possibility of coding the packing pattern of *N* boxes on a binary string with length *N*. A coordinate system with origin at the lower left corner of the pallet is used, with the left lower corner of box i, when packed in the pallet, represented by (x_i, y_i). In the coding, an H-box is represented by the letter H, and a V-box by the letter V. Each box is packed in the feasible position that yields the minimum sum of coordinates, i.e., $x_i + y_i$. In case of ties, the position minimizing y_i is selected. A position is feasible if the box does not overlap with another box packed previously, and is completely packed within the pallet.

Figure IV.7 presents a feasible packing for instance (27, 18, 7, 4), with 16 boxes. The corresponding string is HVHVVVHVVVVVVHVH.

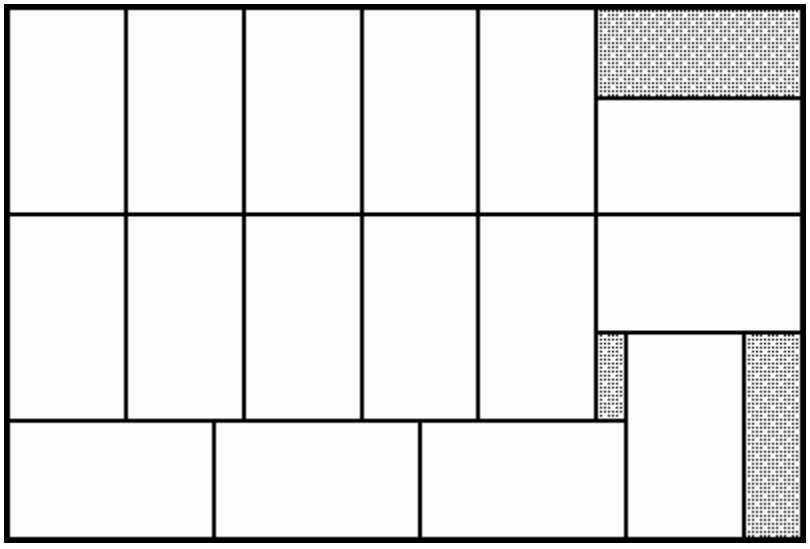

Figure IV.7 *Feasible packing pattern for instance (27, 18, 7, 4).*
The pattern can be represented by the binary string HVHVVVHVVVVVVHVH, with H indicating an H-box and V a V-box.

If every packing pattern could be uniquely represented this way, we could use the coding scheme to create an algorithm with complexity $O(2^N)$ for verifying the feasibility of packing N boxes in an instance of PLP. Although still exponential, this algorithm would be much faster than a general algorithm for 2D-KP, like the one proposed by Murata et al [1995]. Using the bounds on the number of V-boxes and H-boxes obtained using the LP bound, we could reduce this complexity even further.

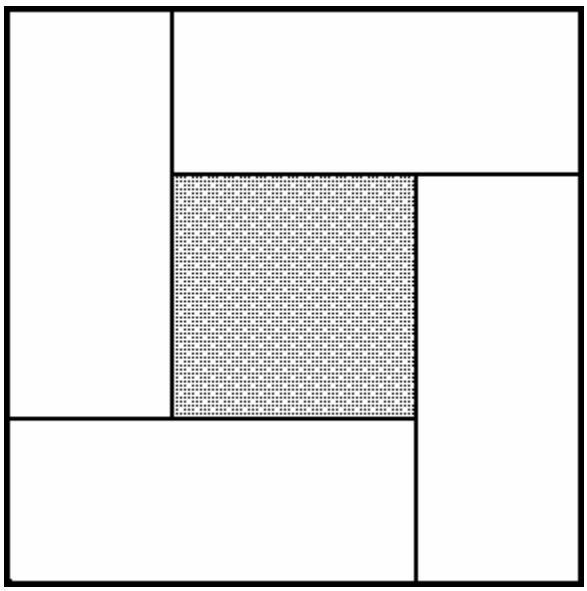

Figure IV.8 *Feasible packing pattern for instance (7, 7, 5, 2).*
This packing pattern cannot be represented with the binary coding scheme.

Unfortunately, there are feasible packing patterns that cannot be represented by this coding scheme. A simple example is observed in instance (7, 7, 5, 2), and the packing pattern depicted in Figure IV.8. The corresponding string would be HVVH, but when trying to pack the third box, the procedure would place it at position (2, 2), not at position (5, 0).

To deal with this situation, we modify our coding scheme to include Z, for zero, representing the position of the left lower corner of a wasted rectangular area. In the example above, the packing pattern is now coded by HVZVH. The Z label occupies the position (2, 2). The next available position is (5, 0), where the next V-box is packed. The only position left to pack an H-box is (2, 5).

The wasted rectangle has length and width at least one unit long, but may have larger dimensions, depending on the use of normal packing patterns. In this case, the wasted area covers the region up to the next positions that could be used to place a box in a normal packing pattern, i.e., positions with the coordinates corresponding to integral combinations of a and b. In the example, the wasted area has dimensions 2×2, because positions (3, 2), (2, 3) and (3, 3) cannot be obtained as a combination of 5 and 2.

As the amount of wasted area is registered whenever a Z label is included in the string, we can use this coding idea in a depth-first search, fathoming the branches of the search tree presenting too much wasted area.

1. Implementation of HVZ Algorithm

The *HVZ Algorithm* is an implementation of depth-first search mentioned above. Each node of the search tree corresponds to visiting the point with minimum sum of coordinate values. The algorithm selects between placing a box, H-box or V-box, or marking the region as wasted. If a box is placed, then all coordinate points covered by the box are labeled as used. If marked as wasted, only one coordinate point is labeled, and the area of the wasted region is added to the total wasted area, *TW*. The next node to be examined corresponds to the next unlabeled coordinate point. If, at any given step of the algorithm, the maximum allowed amount of wasted area, *MW*, is surpassed $(TW > MW)$, then the algorithm backtracks. The algorithm terminates if a feasible packing of N boxes is obtained, or if all possible labels are explored.

The set of coordinate points is defined in the initialization the algorithm, and corresponds to the cartesian product of the sets of feasible partitions of the length and width. In our implementation of the HVZ algorithm, we use two arrays to represent these coordinate points:

- A two-dimensional array, with the first dimension corresponding to feasible partitions of the length, and the second to feasible partitions of the width. This array is used when labeling the coordinate point covered by a box; and
- A one-dimensional array, with pointers to the coordinate points, sorted by sum of coordinate values, and ties solved by smaller width. This array is used to identify the next unmarked point.

After some experimentation with the HVZ algorithm, we identified some strategies, performed at each node, that can yield to shorter run times:

- If the number of boxes loaded is less than $\lfloor N/2 \rfloor$, and the total wasted area is already larger than half of the maximum allowed, $TW > MW/2$, then the algorithm backtracks. This is possible because of the symmetry of the packing patterns. In this case, the complement of the packing pattern contains at least $N/2$ boxes and at most $MW/2$ wasted area, and can be obtained at a later step of the algorithm.
- After placing a box, verify that there is enough space between the box and the borders of the pallet to pack another box. If not, the region is marked as wasted.
- If the optimal packing pattern is formed with smaller blocks packed together, and if each of these blocks can be packed in different ways, then we have several similar, or equivalent, solutions. In order to avoid searching for solutions on patterns similar to others already identified as not optimal, we verify if the box being packed completes a block. If this is the case, we verify if a block with the same dimensions was investigated before, and if the arrangement of this block is the same. If it is not, we backtrack.

2. Results Obtained with the Algorithm

As an empirical performance comparison, we apply the HVZ algorithm to the same instances listed in Table IV.8, and run times are in Table IV.11, also obtained with a Pentium 133 MHz computer. The run times obtained with the G5-heuristic are also included. When the run time exceeded 60 minutes, we stopped the execution of the algorithm. This situation is identified with a dash in the table.

ID	Instance	MSI	N	Run Time (Sec)	
				G5-Heuristic	Exact Algorithm
D1	(22,16, 5, 3)	(22, 16, 5, 3)	23	0.00	0.0
D2	(86, 82, 15, 11)	(23, 22, 4, 3)	42	0.06	0.0
N1	(43, 26, 7, 3)	(43, 26, 7, 3)	53	0.50	1.2
N2	(87, 47, 7, 6)	(87, 47, 7, 6)	97	2.42	–
N3	(153, 100, 24, 7)	(109, 71, 17, 5)	90	0.00	0.2
N4	(42, 39, 9, 4)	(42, 39, 9, 4)	45	0.25	2.0
N5	(124, 81, 21, 10)	(64, 41, 11, 5)	47	0.11	8.0
ST1	(40, 25, 7, 3)	(40, 25, 7, 3)	47	0.19	1.1
ST2	(52, 33, 9, 4)	(52, 33, 9, 4)	47	0.19	3.4
ST3	(57, 44, 12, 5)	(57, 44, 12, 5)	41	0.13	0.8
ST4	(56, 52, 12, 5)	(56, 52, 12, 5)	48	0.21	15.0
ST5	(300, 200, 21, 19)	(127, 85, 9, 8)	149	0.00	–

Table IV.12 *Run times obtained with the HVZ algorithm and the G5-heuristic for selected problems in the literature.*
A dash (–) indicates instances without a computed optimal solution within 60 minutes.

This new exact algorithm is able to obtain solutions to three additional instances from the COVER II set. These instances, shown in Figure IV.9 are (86, 52, 9, 5), (95, 92, 11, 8), and (74, 46, 7, 5).

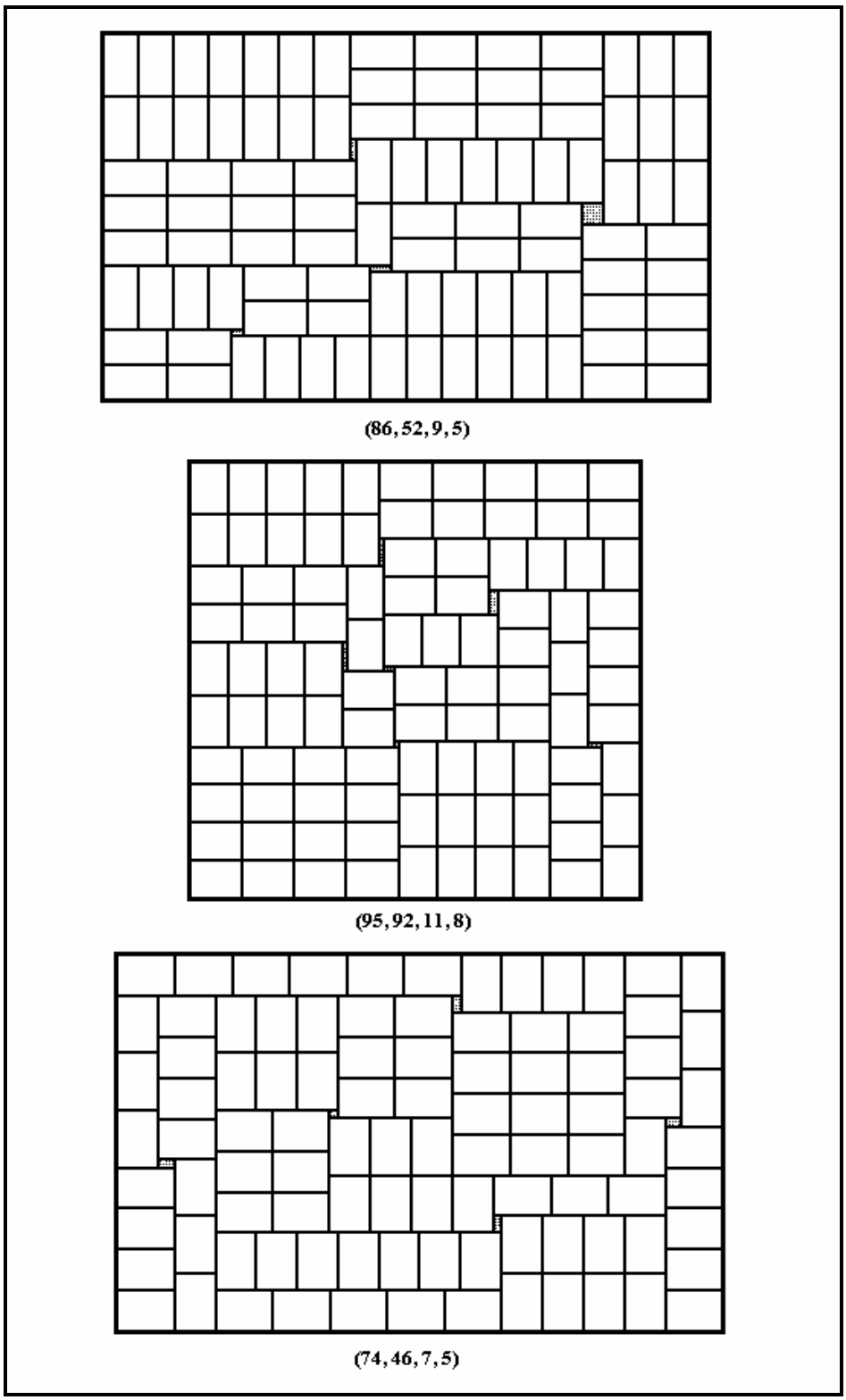

Figure IV.9 *Optimal packing patterns obtained using the HVZ algorithm, for instances (86,52,9,5), (95, 92, 11, 8), and (74,46,7,5).*
These are non 1st-order patterns.

3. Complexity of the HVZ Algorithm

As discussed in the beginning of this section, using a HVZ string to code any packing pattern requires at most $N+W$ characters, where N is the number of boxes packed and W is the wasted area in the optimal packing pattern. Each of the N characters corresponding to packed boxes can take two different values, with 2^N ways to assign these values. If W characters are necessary to represent wasted areas, then there are $\binom{N+W}{W}$ ways of selecting the position of these characters in the string.

It takes $O((N+W)^2)$ to verify if the pattern coded by a given string is feasible. As the number of items packed and amount of wasted area gets larger, the computational effort increases exponentially.

F. A NOT SO EXACT ALGORITHM

Bhattacharya et al [1998] propose an algorithm, *PalDepth*, for PLP, using a new concept they called *Maximal Breadth Filling Sequence* (MBFS). This new concept, based on the idea of efficient partitions, solves PLP by sequentially filling the pallet with boxes forming an efficient partition of the remaining region. The main characteristic is that a partition is always packed with all V-boxes placed after any H-boxes. Different packing strategies are organized in a tree-like structure, and tested depth-first. They show that the algorithm outperforms others in the literature.

When packing the partition (n, m) in the pallet, there can be $\binom{n+m}{n}$ ways of ordering the V-boxes and H-boxes. If only one ordering, with m V-boxes after n H-boxes, is used, then there is a substantial reduction in the solution space. An algorithm exploring this idea can be much faster than others. To justify this approach, the authors present their proof of the validity of only considering this ordering. Their proof is based on showing that any optimal pattern can be converted into a pattern generated by the *PalDepth* algorithm. But their proof is not correct, and it is possible to find optimal solutions to PLP that cannot be generated by their algorithm.

We present a counterexample. In Lemma 1, Bhattacharya et al [1998] claim that given any optimal arrangement for an instance of a PLP, it can be converted to an

arrangement in which the top row contains all V-boxes to the left of all H-boxes. The proof states that a sequential rearrangement of the boxes produces an alternate optimal solution with the necessary arrangement.

In Figure IV.10, we perform the steps established in their constructive proof, until it is not possible to follow it further, and an optimal solution is not obtained. Figure IV.10 (a) shows an optimal arrangement for the instance (43, 26, 7, 3). The first step is to reorganize the top row so that the V-boxes are to the left of the H-boxes, as shown in (b). The process is repeated, as required, for the next rows of boxes, in (c) and (d). But in (d), it is possible to observe that the wasted area is already larger than in the optimal solution. When trying the next step, in (e), we verify that it is not possible to pack the next row of V-pieces. Of course, this transformation could be performed by rotating the pallet vertically, but this step is not considered in their proof. Also, even after this rotation, the PalDepth algorithm cannot produce the resulting packing pattern.

Although Lemma 1 is not true and is used by Bhattacharya et al [1998] to prove the correctness of the PalDepth algorithm, we still haven't shown that their algorithm isn't correct. Using our HVZ algorithm, with the search reduction strategies disabled, we generate all possible optimal arrangements for instance (43, 26, 7, 3). We obtained 12 different solutions, corresponding to rotations and reflections of the whole pallet, and the central hollow block, and none of these could have been achieved with the PalDepth algorithm.

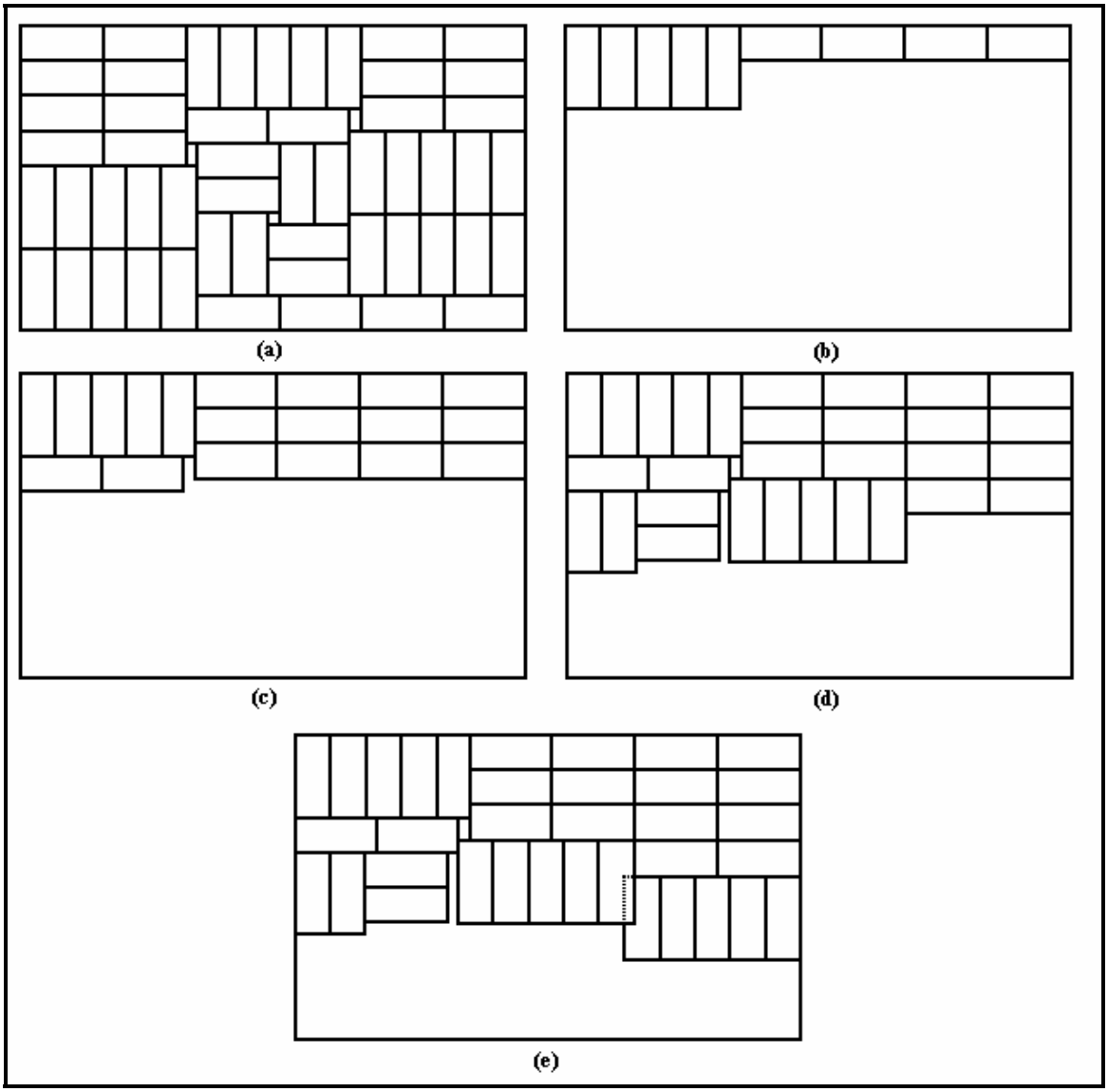

Figure IV.10 *Steps using the procedure proposed by Bhattacharya et al [1998].*
The objective is to convert an optimal pattern for instance (43, 26, 7, 3) into another optimal pattern with all V-boxes in the top row placed to the left of all H-boxes.

V. THE MULTIDIMENSIONAL KNAPSACK PROBLEM

This chapter surveys MD-KP literature with emphasis on exact algorithms accepting non-guillotine cut patterns. It also contains the analysis of a MIP model for 2D-KP, focusing on identifying some of the aspects of model building that affects computational performance.

A. LITERATURE ON SOLVING MD-KP

In this Section, we review existing heuristics, approximation algorithms, and exact algorithm for MD-KP.

1. Heuristics and Approximation Algorithms for MD-KP

Moon and Moser [1967], among other results, present upper bounds on the area *S* of a two-dimensional bin necessary to pack a list *I* of items. This upper bound is based only on the total area of the items, $A = \sum_{i \in I} l_i * w_i$, and the largest side among all items, $D = \max_{i \in I} \{l_i, w_i\}$. If all items in the list are squares, then the list can be packed in any square bin with side $B, B \geq D + \sqrt{A - D^2}$. The list can also be packed in any rectangular bin with $S \geq 2A,$ and shorter side *B*, if $D \leq B$. If items and bin are rectangular, it is shown that the items can be packed in a bin with shorter side *B*, with $D \leq B$, if $S \geq 2A + B^2/2$.

The proof, in all cases, is constructive, and is based on a packing algorithm. The algorithm initially sorts items from largest to smallest, and then packs the items in layers, as in shelves, with the height of each layer defined by the largest item in it. Each item is always placed in the lowest, and leftmost, position that accepts the item, without violating the borders of the bin. Therefore, the algorithm fixes the order in which items are packed, defines where to pack each item, fixes the orientation, and generates only guillotine cut patterns. Figure V.1 presents a typical layout obtained with this approach.

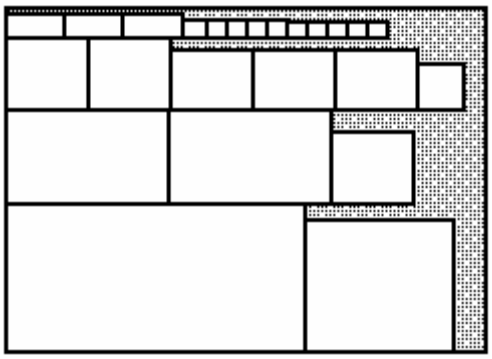

Figure V.1 *Typical packing pattern obtained with the packing algorithm described by Moon and Moser [1967].*

Meir and Moser [1968], using a variation of the same algorithm, improve the upper bound on the area of the bin in the general case, with rectangular items and bins, and show that if $S \geq 2A + B^2/8$, then the set of items can be packed.

George and Robinson [1980] also explore the idea of using layers of items when packing three-dimensional boxes in a container. As in the work of Meir and Moser [1967], items are sorted by size and packed in layers on the length of container. The length of each layer is the length of the first item packed in it. If an empty space is encountered between layers, then a filling scheme is employed in this space. Whenever possible, boxes of same dimensions are packed together. The authors report good practical results, although the worst-case performance is not analyzed.

The algorithm proposed by Coffman et al [1980], *First-Fit Decreasing-Height* (FFDH), for 2D-SPP generates patterns similar to those produced by Moon and Moser [1967] and George and Robinson [1980]. In this case, the dimensions considered by the authors for the bin are width and height. The width of the bin is fixed and the objective is to minimize the height of packing all items. This algorithm obtains heights that are at most 1.7 times larger than the height observed in an optimal solution.

Wang [1983] presents a method for generating solutions for 2D-KP based on the idea of combining items together into larger rectangles and combining these again into larger rectangles, and solves instances with up to 20 items. Further work with this approach includes Oliveira and Ferreira [1990] and Daza et al [1995].

Murata et al [1995] introduce the idea of *Sequence Pairs*, and show that it can be used to represent and solve 2D-KP problems. A *Sequence Pair* representation of the

packing pattern of a set $I = \{a, b, c, d, ...\}$ of items consists of two strings, each with length $|I|$, determined on an oblique grid as shown in Figure V.2.

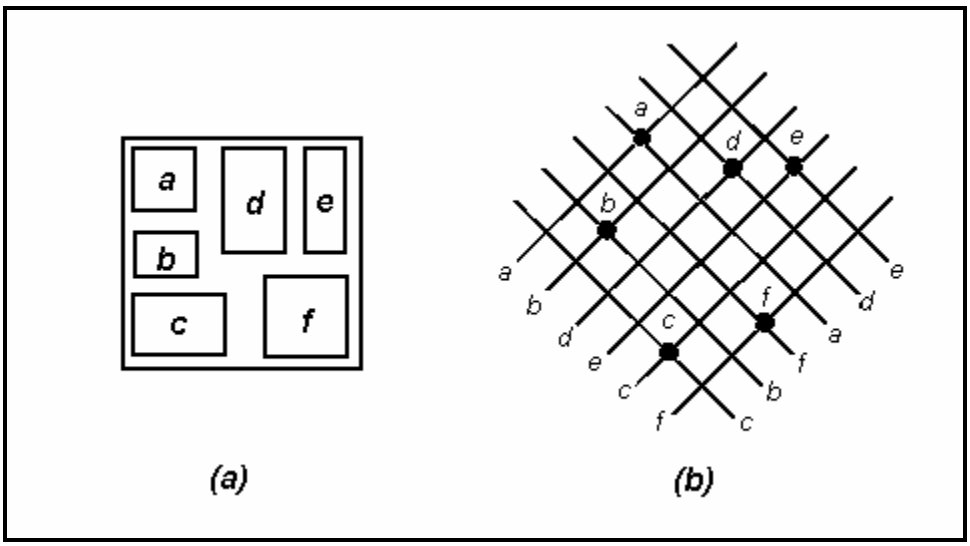

Figure V.2 *Using Sequence Pairs to represent a packing pattern for 2D-KP. The packing pattern in (a), with items {a, b, c, d, e, f} is represented by the sequence pair (a b d e c f, c b f a d e). The order of the symbols in the string determines the position of the item in the oblique grid, and defines the packing pattern (after Kang and Dai [1998]).*

Considering 3D-KP, Li and Cheng [1990] demonstrate that packing strategies as the FFDH have unbounded worst-case performance bounds in 3D-SPP and propose a new approximation algorithm which generates packing with height at most 3.25 times larger than in an optimal solution. Li and Cheng [1992] subsequently study other approximation algorithms.

Chen et al [1995] propose an exact algorithm for container loading based on a MIP model, and show it can be used to solve an instance of 3D-KP with six items.

Scheithauer and Sommerweiss [1998] propose another approach for 2D-KP, not based on layers. Their procedure, the *Four-Block heuristic* based on the G4-heuristic for PLP [Scheithauer and Terno 1996], packs items with identical dimensions in blocks, and uses up to four blocks placed in the corners of the bin. Each block is composed with different items, as in Figure V.3.

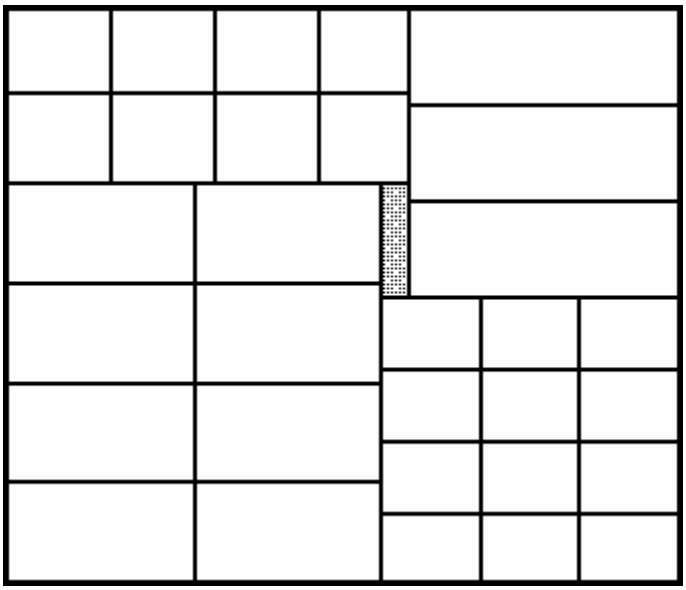

Figure V.3 *Packing pattern generated with the 4-Block heuristic, proposed by Scheithauer and Sommerweiss [1998].*

Scheithauer and Sommerweiss [1998] also discuss some practical conditions influencing the feasibility of packing patterns. These conditions, usually observed in real world applications, are:

Weight**:** The total weight of the items packed in a pallet has to be less than a structural limit.

Placement**:** Some items, because of their density, weight, or contents may not be placed on top of other items.

Splitting**:** If the demand of a given item is large enough to occupy a single pallet, then a pallet packed only with this item has to be used. This restriction has the object of reducing loading costs. This may also apply to a full layer of the same product on a pallet.

Connectivity**:** All items of a given type are to be packed as a block in a pallet. This reduces the number of trips the loader must perform to the storage area.

Stability**:** Packing patterns have to be stable for transport.

Bischoff and Ratcliff [1995] and Adams [1996] address these and other conditions related to container loading.

2. Exact Algorithms for 2D-KP

Gilmore and Gomory [1965], while studying solutions for MD-BPP, are the first to propose an algorithm for solving MD-KP. Noting that "there is no efficient method for solving the generalized knapsack problem of the higher dimensional problem", they concentrate their effort on the version of the problem with two-stage guillotine cutting: the stock rectangle is first slit down its length into strips, and then each individual strip is cut across its width. A third trimming stage is used, if necessary. The orientation of the items is fixed, with no rotations allowed.

Figure V.4 shows the generation sequence of a two-stage guillotine-cutting pattern, with a third trimming stage.

If the application of the two-stage guillotine pattern and fixed orientation restrictions are not dictated by operational conditions, as may be the case in the lumber industry [Gilmore and Gomory 1965], solutions obtained with this approach may "differ" from an optimal solution produced without these restrictions by as much as 100%, as shown in Chapter I, Section E.

Barnett and Kynch [1967] show that if two types of items are to be packed, both with unit width and relatively prime lengths, and if the bin is large enough, then there is a packing pattern with zero wasted area.

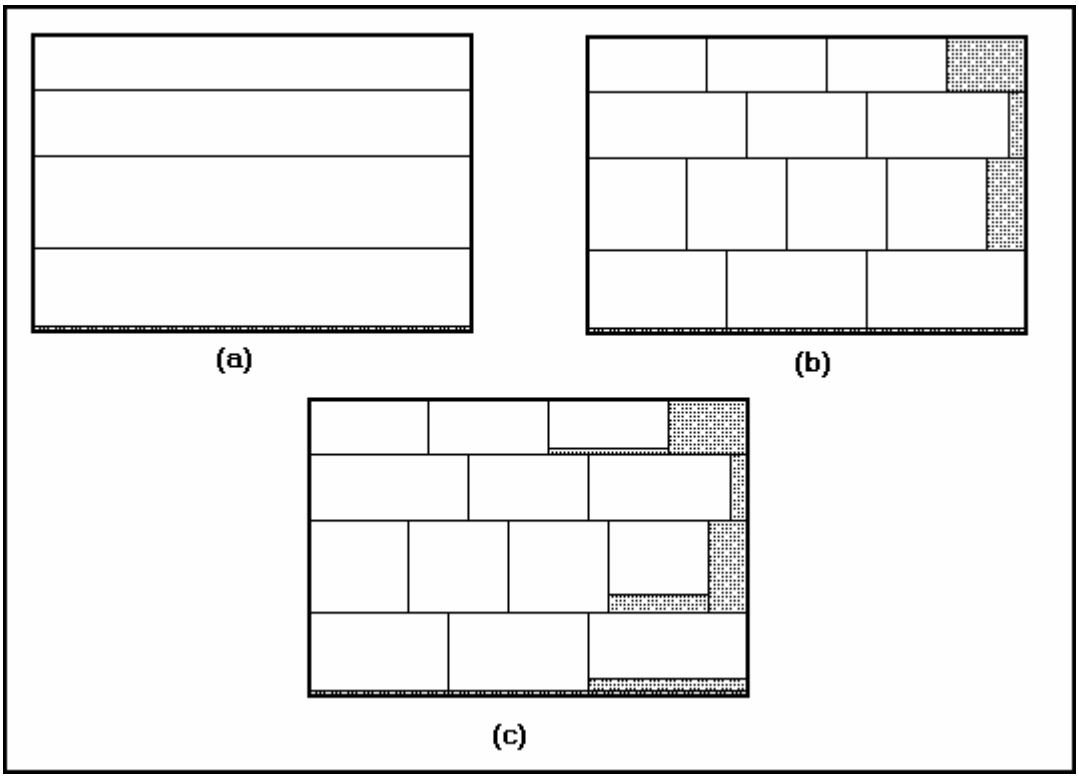

Figure V.4 *Two-stage guillotine cutting pattern studied by Gilmore and Gomory [1965]. (a) The stock rectangle is first slit down its length into strips, and (b) then each individual strip is cut across its width. A third trimming stage (c) is used, if necessary.*

Beasley [1985b] proposes the first exact algorithm for the non-guillotine cutting version of 2D-KP. He formulates the problem as an integer linear program. In his formulation, a large stock rectangle (bin) with sizes $X \times Y$ is to be cut into a set I of different types of items, with piece i ($i \in I$) having dimensions $l_i \times w_i$, value v_i, and a range, (p_i, q_i) for the number of copies packed. The objective is to maximize the total value of the pieces cut. *Normal cutting patterns* are used.

Let $L = \{0 \le r \le X \mid r = \sum_{i=1}^{m} \alpha_i l_i, \alpha_i = 0,...,q_i, i \in I\}$ be the set of possible cut positions in the length, and $W = \{0 \le s \le Y \mid s = \sum_{i=1}^{m} \beta_i w_i, \beta_i = 0,...,q_i, \forall i \in I\}$ be the set of cut positions in the width – the use of normal cutting pattern restricts cut positions to linear combinations of the length and width of the items. The ordered pair (r, s), or (t, u), where $r \in L$, $s \in W$, $t \in L$, $u \in W$, indicates a possible cutting point in the stock. The dimensions of stock

rectangle and items are only used in a pre-process phase, to compute the data used in the model. A model formulation follows.

Indices:

i	Type of items to cut, $i \in I$.
r, t	Cut position in the length, $r \in L, t \in L$.
s, u	Cut position in the width, $s \in W, u \in W$.

Data:

a_{irstu}	1 if piece type *i*, when cut with its bottom left corner at the point (r, s). cuts out the point (t, u) and 0 otherwise.
p_i	Lower bound on the number of items type i.
q_i	Upper bound on the number of items type i.
v_i	Value of item type i.

Variables:

z_{irs}	1 if a piece of type *i* is cut with its bottom left corner at (*r*, *s*) and 0 otherwise.

Formulation:

$$Max \sum_{i \in I} v_i \sum_{r \in L} \sum_{s \in W} z_{irs},$$

subject to

$$\sum_{i \in I} \sum_{r \in L} \sum_{s \in W} a_{irstu} z_{irs} \leq 1, \forall t \in L, u \in W, \quad \text{(V.1)}$$

$$p_i \leq \sum_{r \in L} \sum_{s \in W} z_{irs} \leq q_i, \forall i \in I, \quad \text{(V.2)}$$

$$z_{irs} \in \{0,1\}, \forall i \in I, \forall r \in L, \forall s \in W. \quad \text{(V.3)}$$

Constraints (V.1) guarantee that only one item occupies any given region inside the bin. Constraints (V.2) enforce lower and upper bounds on number of copies of each item.

Using a lagrangean relaxation of this formulation to provide upper bounds, and a specialized branch-and-bound search, Beasley [1985b] solves problems with up to 22 items of 10 different types.

Hadjiconstantinou and Christofides [1995] revisit Beasley's work, and propose a new integer linear program for 2D-KP:

Indices:

i Type of items to be packed, $i \in I$.

j Order of item of type i.

r Cut position in the length, $r \in L$.

s Cut position in the width, $s \in W$.

t Position of the left limit of an unused region, $t = 0,...,X-1$.

u Position of the lower limit of an unused region, $u = 0,...,Y-1$.

Data:

l_i Length of item type i.

w_i Width of item type i.

p_i Lower bound on the number of items type i.

q_i Upper bound on the number of items type i.

v_i Value of item type i.

Variables:

x_{ijr} 1 if the j^{th} copy, $j = 1,...,q_i$, of item i is cut with its bottom left corner at position $r \in L$ and 0 otherwise.

y_{ijs} 1 if the j^{th} copy, $j = 1,...,q_i$, of item i is cut with its bottom left corner at position $s \in W$ and 0 otherwise.

z_{tu} 1 if point $(t,u), t = 0,...,X-1, u = 0,...,Y-1$ has not been cut out by any item i and 0 otherwise.

Formulation:

$$Max \sum_{i \in I} v_i \sum_{j=1}^{q_i} \sum_{r \in L} x_{ijr} ,$$

subject to

$$\sum_{u=s}^{s+w_i-1} \sum_{t=r}^{r+l_i-1} z_{tu} \le (2 - x_{ijr} - y_{ijs}) l_i w_i, \forall i \in I, j = 1,..., q_i, r \in L, s \in W , \quad \text{(V.5)}$$

$$\sum_{r \in L} x_{ijr} \le 1, \forall i \in I, j = 1,..., q_i , \quad \text{(V.6)}$$

$$\sum_{r \in L} x_{ijr} = \sum_{s \in W} y_{ijs}, \forall i \in I, j = 1,..., q_i , \quad \text{(V.7)}$$

$$\sum_{i \in I} w_i \sum_{j=1}^{q_i} \sum_{r=t-l_i+1, r \in L}^{t} x_{ijr} + \sum_{u=0}^{Y-1} z_{tu} = Y, t = 0,..., X-1 , \quad \text{(V.8)}$$

$$\sum_{i \in I} l_i \sum_{j=1}^{q_i} \sum_{s=u-w_i+1, s \in w}^{u} y_{ijs} + \sum_{t=0}^{X-1} z_{tu} = X, u = 0,..., Y-1 , \quad \text{(V.9)}$$

$$x_{ijr}, y_{ijs} \in \{0,1\}, i \in I, j = 1,..., q_i, \forall r \in L, \forall s \in W , \quad \text{(V.10)}$$

$$z_{tu} \in \{0,1\}, t = 0,..., X-1, u = 0,..., Y-1 . \quad \text{(V.11)}$$

Constraints (V.5) set the value of z_{tu} to zero if (t,u) is inside a region occupied by an item. Because of constraints (V.6) and (V.7), each item can be packed at most once in the bin. Constraints (V.8) and (V.9) limit the number of items that can be packed at the same length (V.8) and width (V.9). This formulation does not require the pre-processing phase adopted in Beasley [1985b] and uses a different number of binary variables.

Hadjiconstantinou and Christofides also employ lagrangean relaxation in this model as a bounding procedure within a specialized branch-and-bound algorithm, and solve problems about the same size as those solved by Beasley [1985b].

At each branching decision of the algorithm, a partial packing pattern is defined using a placement rule named *left-most downward placement*. According to this rule, each item is packed at a position "which firstly minimizes the *x*-coordinate of the placement […] and secondly […] minimizes the *y*-coordinate of the placement." A *feasible placement location* is a location where the item cannot be pushed downward and leftward, corresponding to a normal pattern. Their procedure tries packing each item in all feasible

placement locations. Figure V.5 shows admissible placement locations at a possible stage of the branch-and-bound search.

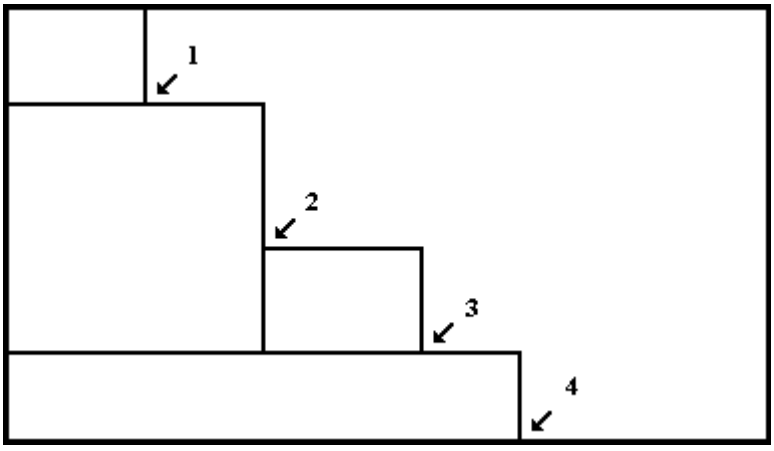

Figure V.5 *Admissible placement locations according to the 'left-most downward placement' rule.*
The next item to be cut (or packed) is positioned in accordance with the rule: first minimize the x-coordinate and then the y-coordinate.

One of the major problems with this rule, as mentioned by the authors, is that different packing sequences can lead to the same packing pattern.

In both approaches described above, binary variables are related to both the number of items and dimensions of the bin (stock piece). The number of binary variables in each model increases with $|I|*|L|*|W|$.

Martello et al [2000] propose a procedure similar to the one proposed by Christofides and Hadjiconstantinou to enumerate patterns for packing individual bins in 3D-BPP. They develop an algorithm, *ONEBIN*, for testing if a given assignment of items to a bin is feasible.

B. A MIP FORMULATION FOR 2D-KP

In this section, we study a MIP formulation for 2D-KP, with the objective of better understanding the effect of using integer and binary variables to represent the relations involved.

One of the main difficulties in formulating this family of problems is representing the physical constraint that no two items share the same region in space. Previous works add an additional constraint of guillotine cuts, or use a discrete grid to represent the

possible packing positions. In most cases, as in Beasley [1985b] and Hadjiconstantinou and Christofides [1995], no rotation is allowed.

The work of Chen et al [1995], proposing a MIP for 3D-BPP, allows rotation, but the objective of the authors is to present "an analytical model to capture the mathematical essence of the problem." They also conclude "a more efficient solution procedure is needed to solve large scale […] problems." They solve an instance of 3D-KP with 6 items, but do not elaborate on the application of the algorithm for 2D-KP.

We implement a similar model for 2D-KP, and use this model to study the effect of some modeling techniques on computational performance. We use, as in Chen et al [1995], the relative position between items to handle the non-overlapping constraints.

Let $x_i\,(y_i), i \in I$, denote the horizontal (vertical) position of the left (lower) side of the item inside the bin, with the origin at the left lower corner of the bin. In our model, we allow 90° rotations.

1. Indicating whether an item is packed in the bin

We use binary variables In_i, for each item $i \in I$, to indicate whether item i is packed in the bin.

2. Indicating rotation

Rotating an item means that the dimensions are interchanged: length becomes width and vice-versa. To indicate whether an item is rotated in a packing, we use a binary indicator variable r_i. This variable takes value 1 if the item is rotated, and its dimensions are interchanged. In mathematical terms, the right upper corner of item i, inside the bin, is given by

$$x_i + l_i(r_i) + w_i(1 - r_i)\,. \tag{V.12}$$

$$y_i + l_i(1 - r_i) + w_i(r_i)\,. \tag{V.13}$$

Another approach is to define two different items, one with the original orientation and the other rotated, with the sizes exchanged. If there exists an upper bound on how many items of each type may be used, each item counts towards this limit.

3. All packed items lie completely inside the bin

When packed in the bin, all items have to be fully contained in the bin, so we use nonnegativity constraints on x_i and y_i, and upper bounds on the position of the right upper corner of the item inside the bin

$$x_i + l_i(1-r_i) + w_i(r_i) \le X, i \in I, \quad \text{(V.14)}$$

$$y_i + l_i(r_i) + w_i(1-r_i) \le Y, i \in I. \quad \text{(V.15)}$$

4. Enforcing no overlapping

For two items to overlap, say i and j, four conditions must be met simultaneously, as shown in Figure V.6:

- The right side of i must be to the right of the left side of j.
- The upper side of j must be above the lower side of i.
- The right side of j must be to the right of the left side of i.
- The upper side of i must be above the lower side of j.

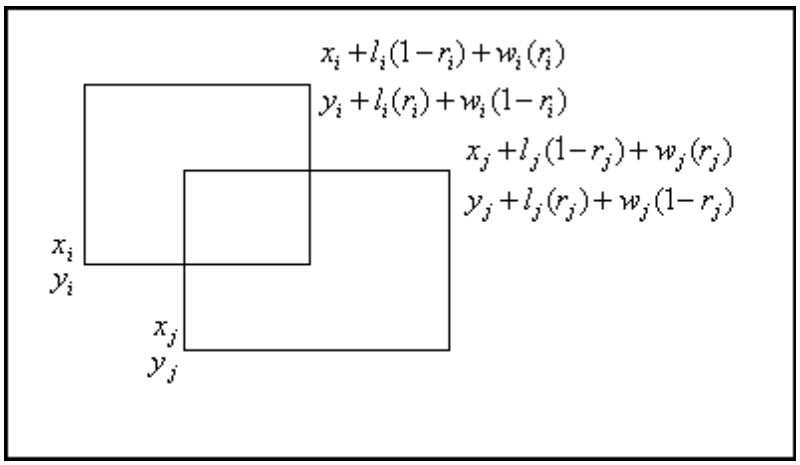

Figure V.6 *This figure represents the overlap of two items, i and j.*
In this example, the right side of i is to the right of the left side of j, the upper side of j is above the lower side of i, the right side of j is to the right of the left side of i, and the upper side of i is above the lower side of j.

When three or fewer of these conditions occur, the items do not overlap. For a better understanding of why this is true, consider each dimension separately. In each of the dimensions, the items define intervals and we can rename the coordinates of the top, or right, side as Max_i, and the bottom, or left, side as Min_i. Then, items i and j overlap in a

given dimension if and only if $Max_i > Min_j$ and $Max_j > Min_i$. This corresponds to two of the previous comparisons in one of the dimensions. Also, items overlap if and only if they overlap in both dimensions.

We use four binary indicator variables px_{ij}, px_{ji}, py_{ij}, and py_{ji} to represent the result of each of the four comparisons, mentioned above, between items i and j, with 1 representing true and 0 false. For example, if the right side of item i is located to the right of the left side of item j, then $px_{ij} = 1$, and if the upper side of j is above the lower side of item i, then $py_{ji} = 1$. We can enforce no overlapping using a *cardinality* constraint, ensuring that at most three of the binary variables take value *1*. Since X and Y are upper bounds on the horizontal (vertical) distance between items inside a container, we define the following set of constraints for the problem as follows:

$$[x_i + l_i(1-r_i) + w_i(r_i)] - x_j \le X * px_{ij}, \quad \forall i \in I, \forall j \in I, j \ne i, \tag{V.16}$$

$$[y_i + l_i(r_i) + w_i(1-r_i)] - y_j \le Y * py_{ij}, \forall i \in I, \forall j \in I, j \ne i, \tag{V.17}$$

$$px_{ij} + py_{ij} + px_{ji} + py_{ji} \le 5 - In_i - In_j, \forall i \in I, \forall j \in I, j > i. \tag{V.18}$$

Constraints (V.16) guarantee that the right side of item i will be to the right of the left side of item j only if px_{ij} is set to 1. There is a similar constraint with the roles of items i and j interchanged. Constraints (V.17) take care of the vertical comparisons. The third set of constraints is the cardinality constraint.

5. Implementing the MIP model

The algebraic formulation is given by

Indices:

i, j Items to be packed, $i \in I, j \in I$.

Data:

X	Length of bin.
Y	Width of bin.
v_i	Value of item $i, i \in I$.
l_i	Length of item $i, i \in I$.
w_i	Width of item $i, i \in I$.

Variables:

px_{ij}	1 if the right side of item j is to the right of the left side of item i and 0 otherwise.
py_{ij}	1 if the top of item j is above the bottom of item i and 0 otherwise.
In_i	1 if item i is packed in the bin and 0 otherwise.
r_i	1 if item i is rotated and 0 otherwise.
x_i	Horizontal position of the left side of item i in the bin.
y_i	Vertical position of the bottom of item i in the bin.

Formulation:

$$Max \sum_{i \in I} v_i * In_i \ , \tag{V.19}$$

subject to

$$x_i + l_i(1-r_i) + w_i r_i \le X, i \in I \ , \tag{V.20}$$

$$y_i + l_i r_i + w_i(1-r_i) \le Y, i \in I \ , \tag{V.21}$$

$$[x_i + l_i(1-r_i) + w_i r_i] - x_j \le X * px_{ij}, \forall i \in I, \forall j \in I \ , \tag{V.22}$$

$$[y_i + l_i r_i + w_i(1-r_i)] - y_j \le Y * py_{ij}, \forall i \in I, \forall j \in I \ , \tag{V.23}$$

$$px_{ij} + py_{ij} + px_{ji} + py_{ji} \le 5 - In_i - In_j, \forall i \in I, \forall j \in I, j > i \ , \tag{V.24}$$

$$px_{ij} \in \{0,1\}, \forall i \in I, \forall j \in I \ , \ py_{ij} \in \{0,1\}, \forall i \in I, \forall j \in I \ ,$$

$$In_i \in \{0,1\}, \forall i \in I \ , r_i \in \{0,1\}, \forall i \in I \ ,$$

$$x_i \ge 0, \forall i \in I \ , \ y_i \ge 0, \forall i \in I \ .$$

Constraints (V.20) and (V.21) enforce the right and top limits of the bin for each item. Constraints (V.22) and (V.23) are used to verify the relative position of items inside the bin. If the items are not packed, we don't care about the values of px_{ij} and py_{ij}. Constraint (V.24) enforces non-overlap between two items only if both items are packed in the bin.

We implemented this model using GAMS [GAMS 2000], and solve some instances from the literature using the CPLEX 6.6 solver [ILOG 2000] and OSL 2 [IBM 2000] on a Pentium IV 1.5 GHz personal computer, with default solution settings. These instances are: twelve instances used by Beasley [1985b], referred to as **NGCUT1** to **NGCUT12**, available from the OR-Library [Beasley 2002]; and instances 3 and 11 from Hadjiconstantinou and Christofides [1995], referred as **HC3** and **HC11**. Instances **NGCUT6** (**HC5**), **NGCUT7** (**HC2**), **NGCUT9** (**HC6**) and **NGCUT12** (**HC7**) are common to both works. Table V.1 lists some details for these instances, including bin sizes, number of distinct types of item, and number of items available. Beasley [1985b] implements his algorithm on a CDC 7600 computer, and Hadjiconstantinou and Christofides [1995] use a CYBER-855 computer. Items are presorted to in an increasing value order.

Instance	X	Y	# Types of Items	# Items	Beasley Run Times (Sec)	H&C Run Times (Sec)
NGCUT1	10	10	5	10	0.9	–
NGCUT2	10	10	7	17	4.0	–
NGCUT3	10	10	10	21	10.5	–
NGCUT4	15	10	5	7	0.1	–
NGCUT5	15	10	7	14	0.4	–
NGCUT6 (HC5)	15	10	10	15	55.2	45.2
NGCUT7 (HC2)	20	20	5	8	0.5	0.0
NGCUT8	20	20	7	13	218.6	–
NGCUT9 (HC6)	20	20	10	18	18.3	5.2
NGCUT10	30	30	5	13	0.9	–
NGCUT11	30	30	7	15	79.1	–
NGCUT12 (HC7)	30	30	10	22	229.0	65.2
HC3	30	30	7	7	–	532.0
HC11	30	30	15	15	–	> 800.0

Table V.1 *Description of instances used by Beasley [1985b] and Hadjiconstantinou and Christofides [1995].*
The fist column contains the references to the instances. The second and third columns contain the length and width of the bin. The forth and fifth columns list the number of different types of items and the total number of items in the instance. The last two columns contain run times reported by the authors, with H&C indicating Hadjiconstantinou and Christofides [1995]. A dash (–) in a cell indicates when an instance is not considered in the work. The execution of instance HC11 was limited to 800 seconds.

We initially address the oriented version of the problems, to be able to compare our results with the previous works. Table V.2 presents run times obtained with our MIP, on a Pentium IV 1.5 GHz personal computer, and the run times reported by Beasley [1985b] and Hadjiconstantinou and Christofides [1995]. In this implementation, no rotations are allowed, so the rotation indicator is removed from the model. Execution times are limited to 1,000 seconds.

Instances	Optimal Solution Value	Beasley Run Times (Sec)	H&C Run Times (Sec)	MIP Solution	MIP Run Times CPLEX 6.6 (Sec)
NGCUT1	164	0.9	–	164	4.0
NGCUT2	230	4.0	–	230	> 1,000
NGCUT3	247	10.5	–	246	> 1,000
NGCUT4	268	0.1	–	268	0.3
NGCUT5	358	0.4	–	358	521.3
NGCUT6 (HC5)	289	55.2	45.2	289	189.5
NGCUT7 (HC2)	430	0.5	0.0	430	0.2
NGCUT8	834	218.6	–	834	> 1,000
NGCUT9 (HC6)	924	18.3	5.2	900	> 1,000
NGCUT10	1,452	0.9	–	1,452	16.2
NGCUT11	1,688	79.1	–	1,688	> 1,000
NGCUT12 (HC7)	1,865	229.0	65.2	1,707	> 1,000
HC3	1,178	–	532.0	1,178	0.2
HC11	1,270	–	> 800.0	1,270	> 1,000

Table V.2 *Run times obtained when solving selected instances from the literature, described in Table V.1.*
The execution of CPLEX 6.6 is limited to 1,000 seconds. In instances NGCUT2, NGCUT8, NGCUT11, and HC11, although the solver attains the optimal solution value, it is not able to verify it. In most instances, an optimal solution is obtained in a few seconds.

CPLEX is able to solve most of these instances from the literature. Only two instances are not solved to optimality within 1,000 seconds, but in another five instances the solver is not able to verify the optimality of the solution. On the other hand, it is able to almost instantly solve instance HC3, which required Hadjiconstantinou and Christofides [1995] more than 500 seconds. A close review of the model reveals the existence of several sources of symmetry, e.g., items of the same type being packed.

Because of the promising results in this first application of the MIP on oriented versions of the instances, we decide to analyze ways of improving the model with the objectives of reducing symmetry and obtaining a tighter LP relaxation of the MIP.

6. Improved MIP model

In order to obtain an improved model, we introduce some simple modifications, including additional constraints and inclusion of new data.

Maximum Area:

The total area covered by all packed items cannot be larger than the area of the bin. This constraint makes the LP relaxation tighter:

$$\sum_{i \in I} In_i * l_i * w_i \leq X * Y . \tag{V.25}$$

Type of item:

The initial formulation does not use the information regarding the type of item being packed – two copies of the same item are considered different items. Therefore, if item i is packed and item j not, but both are of the same type, then we can exchange them and obtain a similar packing pattern. We can reduce this symmetry by including an additional type of data, t_i, type of item i, and use the order in which the items are included in a list to define the sequence to pack items of the same type. The resulting constraint:

$$In_j \leq In_i, i \in I, j \in I, t_j = t_i, j > i . \tag{V.26}$$

Sorting items in the bin:

If multiple items of a given type are packed in the bin, then we can exchange the position of these items and obtain basically the same pattern but with a different order in which the items are packed. To eliminate this symmetry for two items of the same type packed, we force the item with higher order to be packed at a position with larger sum of coordinates:

$$x_j + y_j \geq x_i + y_i, \forall i \in I, \forall j \in I, t_j = t_i, j > i . \tag{V.27}$$

Admissible combinations of indicator variables:

If we consider the indicator variables used to enforce no overlap between two items, px_{ij}, py_{ij}, px_{ji}, and py_{ji}, the admissible combinations, out of the 16 (2^4) possible, are shown in Figure V.7. Item i is fixed at the center while item j takes different positions relative to item i. Each set of four digits indicates the values of the binary variables obtained when performing the comparisons mentioned above. For example, the combination **0011**, on the upper right corner of the picture, indicates:

- The right side of *i* is not to the right of the left side of *j* or $px_{ij} = 0$.
- The top of *i* is not above the bottom of *j* or $py_{ij} = 0$.
- The right side of *j* is to the right of the left side of *i* or $px_{ji} = 1$.
- The top of *j* is above the bottom of *i* or $py_{ji} = 1$.

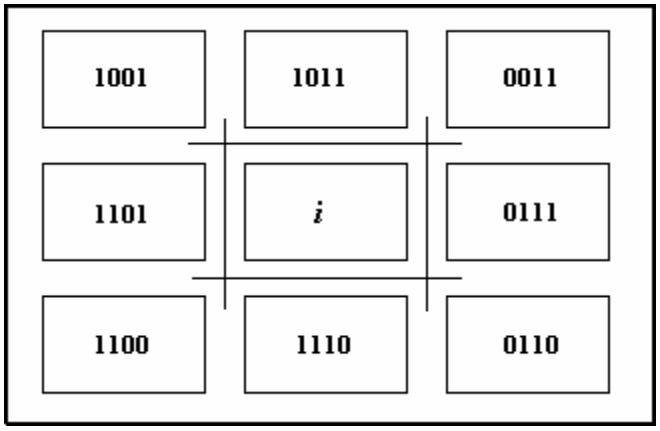

Figure V.7 *Allowed assignments for the indicator variables.*
Although there are 16 possible assignments to four binary variables, there are only nine combinations that could result from the comparison of the relative positions of two rectangles, and eight of those are feasible. The combination 1111 is not feasible and indicates overlap.

From this figure, it is possible to observe that at least two of the binary variables, one for each dimension, are always equal to 1 in a feasible solution. Patterns **0101** and **1010** are inadmissible, because these correspond to one of the dimensions with no binary variables taking value one (not physically possible). We can, therefore, add another set of constraints, forcing the sum of the related indicator variables in each dimension to be at least one:

$$px_{ij} + px_{ji} \geq 1, \forall i \in I, \forall j \in I, j > i, \quad \text{(V.28)}$$

$$py_{ij} + py_{ji} \geq 1, \forall i \in I, \forall j \in I, j > i. \quad \text{(V.29)}$$

Constraints on "large" items:

If two items have the length (width) larger than half of the length (width) of the bin, and both are placed in the bin, then they have to be placed side by side in the width (length) of the bin. In this case, the sum of all items placed in the bin, with length (width) larger than $X/2$ $(Y/2)$, has to be less than the width (length) of the bin.:

$$\sum_{i \in I, l_i > X/2} w_i * In_i \le Y\,, \tag{V.30}$$

$$\sum_{i \in I, w_i > Y/2} l_i * In_i \le X\,. \tag{V.31}$$

Packing the most valued item in the first quadrant:

Given a packing pattern, up to three other patterns might be generated just by reflection and rotation. If several near optimal patterns are present in the instance, the solver requires more time to investigate these patterns. To reduce this symmetry, we select the item with largest value and require it, if packed in the bin, to be placed in the first quadrant:

$$x_i \le X/2, v_i = \max\{v_k \mid k \in I\}\,, \tag{V.32}$$

$$y_i \le Y/2, v_i = \max\{v_k \mid k \in I\}\,. \tag{V.33}$$

We apply this improved model, to the same instances, and the results are listed in Table V.3.

Instances	Beasley Run Times (Sec)	H&C Run Times (Sec)	First MIP Run Times CPLEX 6.6 (Sec)	Improved MIP Run Times (Sec)
NGCUT1	0.9	–	4.0	0.3
NGCUT2	4.0	–	> 1,000	81.8
NGCUT3	10.5	–	> 1,000	6.2
NGCUT4	0.1	–	0.3	0.2
NGCUT5	0.4	–	521.3	0.3
NGCUT6 (HC5)	55.2	45.2	189.5	7.81
NGCUT7 (HC2)	0.5	0.0	0.2	0.2
NGCUT8	218.6	–	> 1,000	> 1,000
NGCUT9 (HC6)	18.3	5.2	> 1,000	21.1
NGCUT10	0.9	–	16.2	0.3
NGCUT11	79.1	–	> 1,000	> 1,000
NGCUT12 (HC7)	229.0	65.2	> 1,000	72.4
HC3	–	532.0	0.2	0.2
HC11	–	> 800.0	> 1,000	> 1,000

Table V.3 *Comparison of the results of the first and improved models. Run times are substantially reduced. In two instances, NGCUT8 and HC11, although the optimal solution is obtained, it is not verified within 1,000 seconds.*

With the improved model producing faster solutions to oriented problems, we adjust the model to solve orthogonal problems. The rotation indicator is inserted in the model, and the "large" item constraint is removed, because of the possibility of rotation. The following constraint concludes the changes in the model:

Only rotate an item if loaded:

We only allow the rotation indicator to be set to 1 if the item is loaded, resulting in the constraint:

$$r_i \leq In_i, \forall i \in I \,. \tag{V.34}$$

Table V.4 presents the results obtained with the orthogonal MIP model, also with a time window of 1,000 seconds.

Instances	# Items	Oriented 2D-KP Optimal Value	Orthogonal MIP Value	Improved MIP Run Times (Sec)
NGCUT1	10	164	193	0.6
NGCUT2	17	230	250	12.5
NGCUT3	21	247	259	124.2
NGCUT4	7	268	268	0.3
NGCUT5	14	358	370	2.1
NGCUT6 (HC5)	15	289	296	> 1,000
NGCUT7 (HC2)	8	430	430	0.2
NGCUT8	13	834	872	> 1,000
NGCUT9 (HC6)	18	924	874	> 1,000
NGCUT10	13	1,452	1,452	1.8
NGCUT11	15	1,688	1,780	> 1,000
NGCUT12 (HC7)	22	1,865	1,794	> 1,000
HC3	7	1,178	1,272	0.7
HC11	15	1,270	1,431	6.95

Table V.4 *Packing values for orthogonal versions of instances from the literature. The first three columns are from Table V.1. The fourth column presents the values obtained with the MIP, and the fifth column the time required by the solver. For instances solved within 1,000 seconds, the value obtained is optimal. For the five instances not solved within the assigned time window, values are not guaranteed to be optimal, and in instances NGCUT9 (HC6) and NGCUT12 (HC7), values are even worse than the results obtained with the oriented version.*

The solver is able to solve to optimality 9 out of 14 instances. In 13 instances, the value returned by the orthogonal MIP is larger than the value obtained with the oriented version. Table V.5 presents details of the solutions for the four instances without a verified optimal solution.

Instances	Orthogonal MIP Value	Integrality Gap (%)	Time to Final Value (Sec)
NGCUT6	296	5.76	14.3
NGCUT8	872	7.57	625.5
NGCUT9 (HC6)	874	10.0	422.3
NGCUT11	1,780	3.4	20.0
NGCUT12 (HC7)	1,794	8.7	182.0

Table V.5 *Details of the solutions at the end of the time window, for instances not solved within 1,000 seconds.*
The first two columns are from Table V.4. The third column presents the value of the integrality gap in the last iteration of the MIP solver, representing the difference, in percentage, between the best integer solution and the best-relaxed solution. The fourth column represents, in seconds, the time required by the solver to identify the solution.

As shown in Table V.5, the largest integrality gap observed is 10.0%, corresponding to an instance where the returned value is smaller than the value obtained with the oriented version.

Although it is not able to obtain an optimal solution within the time window assigned, the orthogonal MIP can solve 2D-KP using a commercial off-the-shelf (COTS) solver without the need of a special tree search.

7. Analyzing the MIP Model

In this section we discuss some of the characteristics of the MIP, in comparison with the integer linear programs from the previous section.

For a given assignment to variables In_i, px_{ij}, py_{ij}, and r_i, we observe that the values on constraints (V.26) are fixed. Constraints (V.22) and (V.23) are reduced to regular upper bounds on variables x_i and y_i. Constraints (V.19) and (V.20) present a $+1$ and a -1 in each row, with independent equations for each dimension. The problem reduces to two independent longest path problems, in an acyclic network, and the position of the items within the bin can be computed efficiently using a network-based algorithm. Items are positioned in the bin according to normal packing patterns. In the work of Beasley [1985b] and Hadjiconstantinou and Christofides [1995], the computation of cut positions corresponding to normal packing patterns is performed in a pre-process phase. In the previous work, normal packing is a pre-requisite, while here it is a result.

As mentioned earlier in this section, in the models proposed by Beasley [1985b] and Hadjiconstantinou and Christofides [1995], the number of binary variables required increases with $|I|*|L|*|W|$. In the present formulation, the number of binary variables increases with $|I|^2$.

Another characteristic is that fixing the relative positions of a subset of the items may cause a significant reduction on the number of feasible combinations left for the indicator variables of other items. As an example, if item *j* is fixed to be to the left of, and above, item *i* (**1001**), and item *k* is fixed to be to the right of, and below, item *i* (**0110**), then item *k* is also forced to be to the right of, and below, item *j* (**0110**).

In a problem with three items, say *i*, *j*, *k*, we use 12 binary variables, just for controlling the relative position among items. If all assignments to these variables are feasible, we have 2^{12} = 4,096 possible assignments. Since there are only eight feasible results when comparing each pair of items, we would have, at most, 8^3 = 512 options. But when the relative positions between items *i* and *j*, and between items *j* and *k*, are given, the range of possible results, when comparing item *i* and *k*, is reduced, as shown in Table V.6. The rows correspond to *i*-*j* comparisons, the columns to *j*-*k* comparisons, and the entries in the table to possible assignments to indicator variables relative to *i*-*k* comparisons. "ALL" means that all eight assignments are feasible. There are only 240 feasible assignments for the 12 binary variables. When a larger number of items are packed in the bin, the reduction can be substantial.

Finally, this number of possible assignments can be reduced even further, if we consider only normal packing patterns. In this case, most of the assignments in Table V.6 will be excluded, because they do not correspond to normal patterns – at most 36 normal patterns exist with three items:

- There are three choices for the item placed in the corner of the bin.
- There are two choices for the next item to pack, and two options of where to place it.
- There are at most three positions to place the last item.

	Comparisons between items j and k								
Comparisons between items i and j		**1011**	**0011**	**0111**	**0110**	**1110**	**1100**	**1101**	**1001**
	1011	1011 1001 0011	1011 0011	1011 0011 0111	1011 0011 0111 0110 1110	ALL	1011 1100 1001 1101 1110	1011 1101 1001	1011 1001
	0011		0011	0011 0111	0011 0110 0111	0011 1110 0110 0111 1011	ALL	0011 1101 1001 1011 0011	0011 1001 1011
	0111			0111 0110 0011	0111 0110	0111 1110 0110	0111 1100 1101 1110 0110	ALL	0111 1001 1101 1011 0011
	0110				0110	0110 1110	0110 1100 1110	0110 1101 1100 1110 0111	ALL
	1110					1110 1100 0110	1110 1100	1110 1101 1100	1110 1001 1011 1101 1100
	1100						1100	1100 1101	1100 1001 1101
	1101							1101 1100 1001	1101 1001
	1001								1001

Table V.6 *Possible results of the comparisons between items i and j, and j and k. Row labels are the comparisons between items i and j. Column labels are the comparisons between items j and k. Table entries are the permissable comparisons between i and k. As an example, if j is above i (**1011**) and k is below j (**1110**), then k can be at any position relative to i (ALL).*

In Chapter VI, we present an algorithm for orthogonal 2D-KP able to identify optimal solutions more efficiently than the MIP model. We also extend this algorithm to solve 3D-KP instances.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. NEW ALGORITHMS FOR MD-KP

This Chapter proposes a new exact algorithm for 2D-KP, based on the HVZ algorithm introduced in Chapter III, and compares the computational performance of this algorithm with some published results for 2D-KP. Then we convert this algorithm, by limiting the number of iterations, into a heuristic, and compare the performance with the exact algorithm and other published results. We also extend the algorithm to solve 3D-KP, and finally introduce a new heuristic for packing a large number of items, but with only a few different types, in a two-dimensional bin.

A. A NEW ALGORITHM FOR 2D-KP

In Chapter I we observe there are four different types of decisions when solving instances of MD-BPP: partition, order, orientation, and relative position. In the case of MD-KP, we also encounter the same basic decisions. The only difference occurs in the partition section, because to solve a MD-BPP we need to determine how many copies of an item to load in which bin, while in MD-KP there is only one bin and we want to maximize the value of the items packed.

The objective function value obtained in an instance of MD-KP depends only on the partition decision – if it is feasible to pack all selected items, then the order, orientation, or relative position among them does not change the value obtained in the packing. Therefore, we can divide the solution of MD-KP in two stages: selecting which items to pack, the *packing list*, and testing whether packing the items in the packing list is feasible.

The HVZ algorithm, proposed in Chapter IV, is developed to verify the feasibility of packing a certain number of boxes of the same type in a pallet. In that context, two characters are necessary to represent boxes at different orientations, and a third character to represent wasted area. But if there are two or more different types of items to pack, we can increase the number of characters used in the coding. If items with types A, B, and C have to be packed in a bin, we can denote $\overline{A}, \overline{B}$, and $\overline{C}$ as rotated items, with the length exchanged with the width. With Z representing wasted area, we can represent any packing pattern produced with these items, and apply the same basic algorithm to verify the

feasibility of a packing list. As an example, if we want to pack the items listed in Table VI.1, selected from instance NGCUT6 [Beasley 1985b], in the 15×10 bin, we obtain the string $D\overline{B}\overline{E}C\overline{F}\overline{A}\overline{G}\overline{H}$, representing the packing pattern in Figure VI.1. In this example, the packing pattern yields no wasted area.

TYPE	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
LENGTH	10	2	11	3	6	4	2	4
WIDTH	3	9	2	8	4	5	4	1
VALUE	74	50	48	46	32	32	11	7

Table VI.1 *Description of items to be packed in a* 15×10 *bin, based on instance NGCUT6 [Beasley 1985b].*

The value of this packing list is 300, and is better than the solution obtained in Chapter V, with the orthogonal MIP limited to 1,000 seconds.

This verification procedure is used within a first stage tree search algorithm to solve 2D-KP. We call this the *Two-Dimensional Diagonal Fill Algorithm (2D-DFA)*.

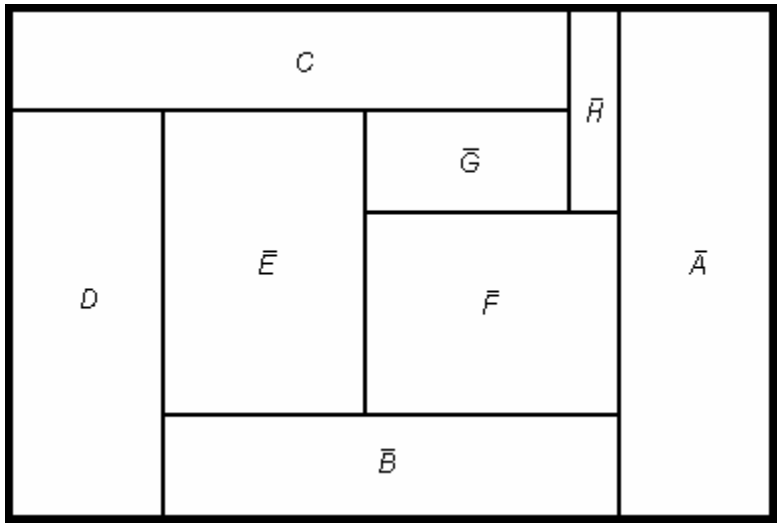

Figure VI.1 *Packing pattern obtained when packing items listed in Table VI.1 in the* 15×10 *bin.*

1. Implementing the 2D Diagonal Fill Algorithm

The 2D-DFA is a branch-and-bound procedure. It divides the list *I* of items to pack into three groups: items examined but not present in the packing list, items included in the packing list, and items yet to be examined. At each node of the search tree, the algorithm selects an item to be included in the packing list. If there exists a feasible pattern with this

packing list, then the procedure branches and an additional item is selected to be included in the list. If it is not feasible to pack the list, then the algorithm backtracks, and the item under consideration is marked as not being used, and the next item is selected.

In order to help reduce the depth of the searchtree, the list I is pre-sorted in decreasing values v_i and, at each node of the search tree, if the sum of the values of all items yet to be examined plus the total value of the present packing list is less than or equal to the best known solution, the algorithm also backtracks. As in PLP case, if less than half of the items in the packing list are packed in the bin, and the packing pattern already presents more than half of the total wasted area, then the algorithm also backtracks.

2. Computational Results

We compare run times and results obtained when applying the 2D-DFA, on an Athlon 1 GHz personal computer, with the algorithms proposed by Beasley [1985b] and Hadjiconstantinou and Christofides [1995], as described in Chapter V. Table VI.2 presents the results for the three algorithms, initially with fixed orientation, because the algorithms proposed in prior work only handle instances with this restriction.

Table VI.2 also contains the results obtained when applying the 2D-DFA to the orthogonal versions of the same instances from the literature. Using the results of the orthogonal model as reference, column "Variation" presents the relative decrease observed in the value of the optimal solution when adopting the fixed orientation restriction. We can observe from the table that a reduction on the order of 15.0% is obtained with instance **NGCUT1**, with an average reduction in the order of 4.9%. The fixed orientation restriction changes the optimal objective function value in 11 out of 14 instances. By allowing items to rotate, the solution space is increased substantially, even when no better solutions are produced, resulting in a considerable increase in run time.

Instance	Fixed Orientation				90º Rotation Allowed		
	Optimum Value	Beasley (Sec)	H & C (Sec)	2D-DFA (Sec)	Optimum Value	Variation (%)	2D-DFA (Sec)
NGCUT1	164	0.9	–	0.0	193	15.0	0.0
NGCUT2	230	4.0	–	0.1	250	8.0	0.1
NGCUT3	247	10.5	–	0.3	259	4.6	0.6
NGCUT4	268	0.1	–	0.0	268	0.0	0.0
NGCUT5	358	0.4	–	0.0	370	3.2	0.0
NGCUT6 (HC5)	289	55.2	45.2	0.3	300	3.7	1.7
NGCUT7 (HC2)	430	0.5	0.0	0.0	430	0.0	0.0
NGCUT8	834	218.6	–	1.4	886	5.9	2.6
NGCUT9 (HC6)	924	18.3	5.2	0.2	930	0.6	10.3
NGCUT10	1,452	0.9	–	0.0	1,452	0.0	0.3
NGCUT11	1,688	79.1	–	1.7	1,786	5.5	84.5
NGCUT12 (HC7)	1,865	229.0	65.2	1.5	1,932	3.5	19.4
HC3	1,178	-	532.0	0.0	1,272	7.4	0.0
HC11	1,270	-	800.0	0.0	1,431	11.3	0.6

Table VI.2 *Comparison of run times and results obtained with the 2D-DFA.*
This table compares results and run times obtained with the 2D-DFA with other algorithms, for oriented instances of 2D-KP in the literature; and presents results and run times of the 2D-DFA, when applied on the same instances, but with 90º rotations allowed. The column variation indicates the decrease, in percentage, resulting from restricting the instances of 2D-KP to fixed orientation.

3. A Variation of the 2D-DFA

The 2D-DFA can be implemented with a different first stage. In the procedure described above, whenever an item is added to the group of items being packed, the algorithm verifies that a feasible packing pattern exists using these items. But if we have a good solution, previously obtained with the algorithm or a heuristic, we could decide to only verify the feasibility of packing patterns with total value larger than the current best value. In this case, we would have to solve fewer feasibility problems. This variation to the 2D-DFA is also implemented, but the differences on computational performance are small.

B. A NEW 2D-KP HEURISTIC – LIMITING THE NUMBER OF ITERATIONS

Although the 2D-DFA is able to solve some instances of 2D-KP from the literature, the algorithm's worst-case exponential run time results in extremely long run times as soon

as the number of different types of items, or the number of items of each type available, gets larger. In this case, as previously discussed, we need to use a heuristic or to add additional restrictions in order to obtain a good solution in a timely fashion.

Wang [1983] adopts the second alternative, adding the restriction that the patterns are of guillotine type. Her algorithm minimizes the trim loss, or the amount of wasted area, in constrained 2D-KP instances, constructing larger rectangles by consecutively joining items. Daza et al [1995] use "informed methods" to improve Wang's algorithm, and compare their algorithm with two others from the literature – Wang's [1983] and Oliveira and Ferreira [1990] - using eight 2D-KP instances, **P-1** to **P-8**, described in the paper. We apply the 2D-DFA to the same instances. Table VI.3 presents the results, and run times, from Daza et al [1995] and obtained by 2D-DFA.

Instance	Daza et al [1995]		2D-DFA	
	Trim Loss	Run Time (Sec)	Trim Loss	Run Time (Sec)
P-1	0	9.33	0	0.00
P-2	29	48.94	22	115.07
P-3	43	109.74	43	1,033.14
P-4	31	38.01	31	1,342.05
P-5	0	1.32	0	0.00
P-6	0	8.12	0	0.11
P-7	8	13.34	5	33.23
P-8	34	21.53	34	274.53

Table VI.3 *Trim loss and run times presented in Daza et al [1995] and obtained by 2D-DFA for eight instances of 2D-KP.*
The 2D-DFA is executed on an Athlon 1 GHz personal computer.

Because the 2D-DFA considers patterns other than those generated by guillotine cuts, it is able to find solutions to two instances with less waste than the algorithm proposed by Daza et al [1995], but at the expense of a considerable increase in run time.

We investigate an alternate approach: the *Limited Iteration 2D-DFA* (*2D-LDFA*) limits the number of backtracks in the verification stage of the algorithm. If the feasibility of a packing list is unknown when this limit is reached, then it is considered as infeasible. We have no way to specify this limit a priori, with the number of items in the list and the size of the bin being important in the selection. If this limit is set too small, solutions might be far from optimal. If the limit is too large, long run times result.

Table VI.4 has the results of the application of two different levels: 1,000 and 10,000 backtracks allowed. With the former level, the optimal solution is obtained in 5 out of 8 cases, taking on average 11.79 seconds to solve each instance. With the latter level, all instances are solved to optimality, taking on average 37.01 seconds.

Instance	2D-LDFA – 1,000 Backtracks		2D-LDFA – 10,000 backtracks	
	Wasted Area	Run Time (Sec)	Wasted Area	Run Time (Sec)
P-1	0	0.00	0	0.00
P-2	24	31.58	22	99.20
P-3	87	15.49	43	90.41
P-4	31	16.15	31	86.01
P-5	0	0.00	0	0.00
P-6	0	3.62	0	0.11
P-7	5	6.87	5	27.96
P-8	70	20.60	34	65.69

Table VI.4 *Wasted areas and run times observed using the 2D-LDFA, with limits on 1,000 and 10,000 backtracks on the verification stage of the algorithm.*

C. EXTENDING DFA TO SOLVE 3D-KP

A natural step to follow is to extend the DFA algorithm to solve three-dimensional problems. In this section, we implement an algorithm for 3D-KP, verify its performance with some test instances, and propose a simple heuristic based on limiting the number of iterations in the verification stage of the algorithm.

1. Implementing the Three-Dimensional Diagonal Fill Algorithm

The transformation of the DFA to work with three-dimensional problems is straightforward. No changes are required in the first stage of the algorithm, because it only selects items to be loaded in the bin. This stage is essentially independent of the dimensionality of the problem. The second stage requires the addition of a third dimension, the height, in the grid representing the positions in the bin where items may be positioned in a normal packing pattern, and new labels to represent all six possible orientations of items inside the bin. We call this algorithm the *Three-Dimensional Diagonal Fill Algorithm – 3D-DFA*.

2. Computational Results

The 3D-DFA is applied to the same instance presented by Chen et al [1995], and it computes an optimal solution in less than a millisecond. Then, we generate a set of 10 random instances, with the following characteristics:

- All bins have dimensions $30 \times 20 \times 15$.
- Items dimensions are selected, independently, in the range [5, 15], with uniform probability.
- Item values are integral values based on the volume of the corresponding item type, and are selected, also with uniform probability, in the range $[0.05 * ItemVolume, 0.15 * ItemVolume]$.
- The number of items available from each type are selected in {1, 2, 3}.
- Items are generated and included in an instance until the sum of item volumes is larger than a number generated between 7,200 and 10,800, or between 80% and 120% of the volume of the bin.

Table VI.6 contains general information of the instances **M-1** to **M-10**, generated above. The information, presented in column order, is: instance, number of different types of item, number of items, total volume of items, total value of the items and an upper-bound on the value of the optimal packing, obtained by modeling the problem as a 1D-KP instance, using the volumes and values.

Instance	# Item Types	# Items	Total Volume of Items	Total Value of Items	Upper-Bound on Optimal Value
M-1	6	15	13,411	1,281	941
M-2	8	17	10,840	1,116	990
M-3	2	6	8,352	1,221	1,221
M-4	4	10	10,074	1,060	909
M-5	6	13	10,120	1,022	905
M-6	7	16	12,065	1,107	917
M-7	6	10	9,889	824	752
M-8	2	5	12,468	847	579
M-9	5	10	11,511	1,158	971
M-10	6	14	12,567	958	767
Averages	5.2	11.6	11,129.7	1,169.4	895.2

Table VI.5 *Descriptive information of the random instances used to investigate the performance of the 3D-DFA.*
The information, presented in column order, is: instance, number of different types of item, number of items, total volume of items, total value of the items and an upper-bound on the value of the optimal packing, obtained using the volumes and values in as an 1D-KP instance

Table VI.6 presents the results, and run times, observed when applying the 3D-DFA on the 10 random instances, M-1 to M-10, oriented and orthogonal versions.

Instance	Oriented		Orthogonal	
	Value	Run Time (Sec)	Value	Run Time (Sec)
M-1	663	3.25	869	741.28
M-2	843	44.71	939	3,222.39
M-3	814	0.00	946	0.00
M-4	666	0.10	892	3.35
M-5	796	0.82	905	69.20
M-6	677	5.44	789	1,941.40
M-7	616	0.00	720	6.21
M-8	386	0.00	386	0.00
M-9	756	0.00	883	5.05
M-10	590	0.27	712	77.28

Table VI.6 *Optimal values and run times, in seconds, obtained with the 3D-DFA, oriented and orthogonal versions, on 10 random instances.*

The optimal values obtained in the oriented version are, on average, 14.5% smaller than those generated with the orthogonal version. In one case, the fixed orientation

restriction accounts for a 25.3% decrease in optimal solution value. But, as expected, larger problems require much longer periods of time before an optimal solution is obtained. As in the two-dimensional version, the use of a heuristic may be the only approach to generate "good" solutions in a timely fashion.

D. THE LIMITED ITERATION 3D-KP HEURISTIC

As in the two-dimensional case, the 3D-DFA can be converted into a heuristic by limiting the number of iterations on the verification stage of the algorithm. Table VI.8 presents the comparison of results obtained with the orthogonal 3D-LDFA with the upper bound based on 1D-KP, and the results of the oriented and the orthogonal 3D-DFA.

Instance	Upper Bound	3D-DFA		3D-LDFA	
		Fixed Orientation	90º Rotations Allowed	Value	Run Time (Sec)
M-1	941	663	869	869	40.43
M-2	990	843	939	897	54.43
M-3	1221	814	946	946	0.00
M-4	909	666	892	822	1.70
M-5	905	796	905	866	13.13
M-6	917	677	789	759	109.96
M-7	752	616	720	665	1.10
M-8	579	386	386	386	0.00
M-9	971	756	883	883	1.54
M-10	767	590	712	704	39.27

Table VI.7 *Packing values, and run times, obtained with the 3D-LDFA.*
Execution is limited to 200 backtracks before restarting the search tree, at the first level of the tree, on an Athlon 1 GHz personal computer.

E. A FIVE-BLOCK HEURISTIC FOR 2D-KP

In Section C we present a heuristic for 2D-KP based on limiting the number of iterations of the verification stage of the DFA. There is no constraint on the number of items of the same type in the packing pattern, so if this number is large, even the proposed heuristic can require long run times.

There is a variation of the 2D-KP for which the use of a PLP heuristic, combined with partitioning the bin in blocks, may yield very good results. Scheithauer and Sommerweiss [1998] investigate the *Rectangle Packing Problem* (RPP). In an instance of

RPP, a relative large amount of items, from a few different item types, have to be packed in a bin, with the objective of maximizing the used area. Using the typology proposed by Dyckhoff [1990], this problem is of type $2/B/O/F$. There may be constraints on the minimum and maximum number of items of each type in the packing pattern.

In their work, the authors divide the bin in up to four blocks, as in the Four-Block heuristic for PLP, and use the G4-heuristic [Scheithauer and Terno 1996], within each block, to select the item type to pack in the block. Only items of the same type are packed in each block. The wasted area computed up to a step in the algorithm is used as a bounding rule. The authors report that the heuristic, the *Four-Block Heuristic*, yields area usage above 99% of the bin area in several random instances solved. One of the possible improvements considered by the authors, but not implemented, is the use of five blocks, with the addition of a central block, as in the five-block heuristic for PLP.

Because the analysis of the G5-heuristic for PLP, in Chapter IV, shows that it generates solutions as good as the G4-heuristic, we investigate the *Five-Block Heuristic* for 2D-KP using the G5-heuristic to define the composition of each block.

1. Implementing the Five-Block Heuristic for 2D-KP

The main difference between the four-block and five-block heuristics, as indicated by their titles, is the maximum number of blocks containing items of the same type allowed in the solution. The four-block heuristic generates solutions similar to Figure II.7, while the five-block heuristic can produce solutions resembling Figure II.8.

As in the PLP case, we only consider linear combinations of the length and width of each item being packed when defining the dimensions of the two initial corner blocks (blocks I and III). Then, we apply the G5-heuristic within each block to determine the packing pattern. We store the best packing patterns obtained for a block with given dimensions so we can use it again later in the algorithm, if necessary.

For each item k, we consider, without loss of generality, that $l_k \geq w_k$. The dimensions of block i, $i = 1,2,3,4,5$, are given by (L_i, W_i), and C_i the total value of the items packed in block i. In the algorithm proposed by Scheithauer and Sommerweiss

[1998], C_i is defined by the used area within each block. In the present algorithm, we consider the more general case of 2D-KP.

Let $G_l = \{g \mid g \le L, g = n * l_k + m * w_k, \textit{for some item } k\}$ be the set of dimensions to be investigated as the length of blocks I and III.

Let $G_w = \{g \mid g \le W, g = n * l_k + m * w_k, \textit{for some item } k\}$ be the set of dimensions to be investigated as the width of blocks I and III.

The pseudocode for the five-block heuristic for 2D-KP is:

For each $L_1 \in G_l$, $L_2 \leftarrow X - L_1$

For each $W_1 \in G_w$, $W_3 \leftarrow Y - W_1$, compute C_1

For each $L_3 \in G_l$, $L_4 \leftarrow X - L_3$, compute C_4

For each $W_3 \in G_w$, $W_2 \leftarrow Y - W_3$, compute C_2 and C_3

If no blocks overlap, then

Compute L_5, W_5 and C_5

If $C_1 + C_2 + C_3 + C_4 + C_5$ is higher than the previous best, record solution.

2. Computational Results

Scheithauer and Sommerweiss [1998] use sets of 100 random generated instances to investigate the performance of the four-block heuristic. The bin has fixed dimensions, $1,250 \times 800$, and items are uniformly generated in the range $[100, 300]$. All instances initially have four different item types, and additional item types are added until ten different types are available in each instance. Then the algorithm is applied successively to each instance, with from four up to ten item types. In our analysis of the performance of the five-block heuristic, we use a set of instances generated using the same procedure. Initially, the objective is to minimize wasted area.

Table VI.8 presents some statistics computed from the results of the application of the five-block heuristic to the set of 100 instances. Columns are numbered from four to ten, representing the number of different item types in each instance group. The "G5-heuristic" and "Four-block" rows present the average area usage resulting from the selection of the best G5-heuristic pattern, and best four-block pattern. The "Five-block" row has average

area usage obtained with the five-block, with the same instances. The number of instances, out of 100, where the five-block yields a better result than the four-block is counted in the row "Better."

	4	5	6	7	8	9	10
Four-block (%)	98.70	98.94	99.15	99.27	99.39	99.46	99.52
G5-heuristic (%)	95.72	96.22	96.56	96.78	97.00	97.15	97.32
Five-block (%)	98.82	99.04	99.23	99.34	99.46	99.51	99.57
Better	36	35	36	42	39	35	39

Table VI.8 *Average area usage obtained with three different block heuristics. Columns are numbered from four to ten, representing the number of different item types in each set of instances. Each set is composed of 100 randomly generated problems. The "G5-heuristic" and "four-block" rows contain the average area usage resulting from the selection of the best G5-heuristic pattern, and best four-block pattern. The "Five-block" row has average area usage obtained with the five-block, with the same instances. The number of instances, out of 100, where the five-block yields a better result than the four-block is counted in the row "Better."*

In the table, we can observe that more than 35% of the instances had a better solution using the five-block heuristic, although the improvement in area usage is small. The relative change in wasted area, when it occurs, is on the order of 20%. The authors of the four-block heuristic observe that up to 96 items can be packed in the instances considered. In this case, the items are small relative to the bin, making them easer to pack.

The average run times for the five-block heuristic are compatible with those reported for the four-block heuristic, but the run times with the five-block heuristic do not increase with number of types of items as quickly as with the four-block (as reported by Scheithauer and Sommerweiss [1998], and shown in Table VI.9). The run times presented in the table for the five-block corresponds to the execution on a Athlon 1 GHz personal computer. For the four-block heuristic, times are those reported in the original work on a 586 200 MHz personal computer.

Heuristic	4	5	6	7	8	9	10
Four-block (Sec)	0.20	0.34	0.54	0.73	1.02	1.37	1.78
Five-Block (Sec)	0.10	0.14	0.17	0.21	0.24	0.28	0.34

Table VI.9 *Average run times, in seconds, with the five-block heuristic, in the present research, and the four-block heuristic, as reported by Scheithauer and Sommerweiss [1998].*

We generate a second set with 100 random instances, this time with items with dimensions selected in the range $[200, 400]$, and the same bin. Instances with this range of

items contain a minimum of 6 and a maximum of 24 items. Table VI.10, with the same layout as Table VI.9, presents the results for this new set.

	4	5	6	7	8	9	10
Four-block (%)	96.41	97.08	97.47	97.80	98.10	98.39	98.52
G5-heuristic (%)	92.46	93.24	93.65	93.91	94.28	94.61	94.80
Five-block (%)	96.51	97.19	97.52	97.90	98.21	98.49	98.60
Better	11	14	12	18	25	22	23

Table VI.10 *Results obtained with 100 instances, item dimensions selected in the range [200,400], with the same layout as Table VI.8.*

In this range, although the number of cases in which the five-block heuristic generates a better solution is smaller than in Table VI.9, the relative change in the wasted area is about the same.

3. A More General Case of 2D-KP

Besides the number of blocks, and the underlying PLP heuristic, another difference between the four-block heuristic [Scheithauer and Sommerweiss 1998] and the proposed five-block heuristic is that the former is implemented to solve the variation of 2D-KP, based on maximizing the area usage of packed items, or minimization of trim loss. The four-block heuristic proponents do not directly address the more general case, where the value v_i of item type i is different than, and not even related to, the area of the item.

The five-block heuristic is developed with this more general instance in view, and it can select the best block composition based on the total value of items packed in the block, not the wasted area in it. But in this case, the bounding procedure based directly on wasted area does not work, because it is possible to generate a block with more wasted area, but with larger value.

To overcome this difficulty, we adopt a different bounding procedure:

- Compute, a priori, the Maximum Value Density (MVD) – the largest ratio between the value of an item and its area: $MVD = \underset{i \in I}{Max}\left(v_i/(l_i * w_i)\right)$;
- After defining the dimensions of a block, and solving the associated PLP instance, add all wasted area observed up to that point, including any inter-block wasted areas;

- Compute the useable area left in the bin, subtracting all wasted area from the bin area;
- The sum of the values of the blocks already defined, plus the product of the MVD and useable area is an upper bound on total number of items; and
- If this sum is smaller than the present best solution value, then move to another branch of the search tree.

This bounding procedure is not as efficient as the procedure used when the problem involves only the minimization of trim loss, where the value density is just equal to one.

To verify the performance with this new heuristic, we generate another set, with the same characteristics as the set applied in the minimization of trim loss case, but with items with value densities uniformly drawn in the range $[0.5,1.5]$. Table VI.11 contains the average ratio of packing value per bin area obtained with the best G5-heuristic solution and with the five-block heuristic, for each group of 100 instances. The table also includes the number of instances, out of 100, where the five-block heuristic yields a better solution then the G5-heuristic. The last row has the average run times for the five-block, on an Athlon 1 GHz personal computer.

	4	**5**	**6**	**7**	**8**	**9**	**10**
G5-heuristic (%)	1.17	1.22	1.24	1.26	1.27	1.28	1.29
Five-block (%)	1.20	1.25	1.28	1.30	1.32	1.33	1.34
Better	64	79	82	85	86	88	89
Run Times	0.39	0.62	0.84	1.18	1.66	2.10	2.71

Table VI.11 *Average ratio of packing value by bin area obtained with the G5- and five-block heuristics.*
Columns are numbered from four to ten, representing the number of different item types in each set of instances. Each set is composed of 100 randomly-generated problems. The "G5-heuristic" and "Five-block" rows have average ratio of packing value by bin area obtained with the corresponding heuristic. The number of instances, out of 100, where the five-block yields a better result than the four-block is counted in the row "Better." Average run times, in seconds, observed with the five-block heuristic are listed in the last row.

VII. THE MULTIDIMENSIONAL BIN PACKING PROBLEM

Martello et al [2000] present an exact algorithm for 3D-BPP, and state "to our knowledge, no algorithms for the exact solution of 3D-BPP have been published." This work comes after more than three decades dedicated by several researchers to this problem, since Gilmore and Gomory [1965] first described their formulation of the *Three-Dimensional Cutting Problem*. But Martello et al had to give up something in order to control the exponential growth of the enumeration problem – they only consider instances with fixed orientation.

In the previous chapters, we have developed the necessary tools to determine if it is feasible to pack a given list (packing list) of items in a single bin, and to select the packing list with maximum value to pack in a single bin in both two- and three-dimensions. To conclude our studies in this dissertation, we employ these tools to develop exact and heuristic algorithms for 2D-BPP and 3D-BPP, which handle orthogonal problems (only 90º rotations are allowed), and show that these algorithms can be used to solve instances of MD-BPP from the literature.

A. SOLUTION APPROACHES FOR MD-BPP IN THE LITERATURE

The first solution approach for 2D-BPP in the literature is proposed by Gilmore and Gomory [1965]. It involves the combined use of Two-Stage Guillotine Cutting Patterns (shown in Chapter IV), linear programming with column generation techniques (previously applied in the one-dimensional version [Gilmore and Gomory 1961]), and the solution of two knapsack problems, one for each dimension, at each column generation step. As in the one-dimensional case, fractional solutions are rounded to integer ones. The authors also present and discuss the extension of their solution technique to 3D-BPP.

Chen et al [1995] propose a different approach, based on a MIP for 3D-BPP. Only a small instance of 3D-KP is solved in the paper. As discussed in Chapter V, adopting a MIP model and a general-purpose solver sacrifices information regarding the geometric construction of the constraints.

Martello et al [2000] analyze new bounds for 3D-BPP, and present both exact and approximate algorithms. Their exact algorithm is based on a two-level decomposition, also used for 2D-BPP [Martello and Vigo 1998], and can be traced back to the original ideas of Gilmore and Gomory [1961]. Both works consider only fixed-orientation instances.

Wang [1983] proposes an approximation algorithm for *The General Cutting Stock Problem*, a version of orthogonal 2D-BPP, allowing bins of different sizes but considering only guillotine cuts. Chung et al [1982] uses a hybrid procedure, based on 2D-PP, in an approximation algorithm for 2D-BPP, also with fixed orientation.

Heuristics for solving 3D-BPP are proposed by Ivancic et al (as reported by Bischoff and Ratcliff [1995]), Bischoff and Ratcliff [1995], Terno et al [2000], Lodi et al [2002b], among others.

B. OBTAINING EXACT SOLUTIONS TO MD-BPP

Our algorithm to solve MD-BPP is a *Branch-and-Price* (*B&P*) algorithm (e.g., Johnson et al [2000]), based on the original cutting stock formulation of Gilmore and Gomory [1961], but solved to optimality. Our branch-and-bound algorithm uses a linear relaxation of Gilmore and Gomory's integer linear program with dynamic column generation, employing our proposed two-stage algorithms for MD-KP for column generation.

As described by Gilmore and Gomory [1961], given a list I of item types to cut, with demand $d_i, i \in I$, if set of all feasible cutting patterns, J, is known beforehand, our problem becomes selecting how many times each cutting pattern is used to produce the optimal solution, attending the demand of all items, and using the least amount of stock. Let $a_{ij}, i \in I, j \in J$ represent the number of items of type i in the pattern j. If two patterns present the same numbers of each item type, even in a different layout, they are considered to be the same pattern.

The formulation is:

Indices:

i Item types, $i \in I$.

j Cutting patterns, $j \in J$.

Data:

d_i Demand of item type i.

a_{ij} Number of items of type i in pattern j.

Variables:

x_j Number of times pattern j is used in the cutting process.

Formulation:

$$Min \sum_{j \in J} x_j ,$$

subject to

$$\sum_{j \in J} a_{ij} x_j = d_i, i \in I , \qquad \text{VII.1}$$

$$x_j \in Z^+ .$$

The main deficiency with this approach is that the feasible patterns are not known beforehand, because identifying all potential patterns requires too much computational effort. To overcome this difficulty, Gilmore and Gomory [1961] propose the application of the simplex algorithm, and the generation of a pattern only when it prices favorably. In the one-dimensional case, generating a pattern requires solving a 1D-KP [Gilmore and Gomory 1961]. In the two-dimensional case, when considering only two-stage cutting patterns, two instances of 1D-KP are solved to generate a pattern [Gilmore and Gomory 1965]. In the general case, addressed in this dissertation, a 2D-KP or 3D-KP is solved to generate a pattern.

The algorithm starts by generating the columns necessary to establish a simplex basis and computing the simplex multipliers associated with it. The basis is obtained by solving an instance of MD-KP considering only one type of item per column generated, and the multipliers are used to identify a favorable new column, again using the algorithm for MD-KP. A new basis and corresponding simplex multipliers are obtained, and the search for a new column is repeated. When no additional favorable columns can be generated, the algorithm proceeds to the branch-and-bound stage. In this stage, as new integrality restrictions are generated, each new extended linear program instance is solved, again, with column generation.

To investigate this new algorithm, we implement it in a non-automated fashion. Once the first set of columns, corresponding to the initial relaxed LP is generated, this set is exported to an EXCEL 2000 [Microsoft 1999] spreadsheet, where the IP solver is used to identify the cuts to be added to the relaxed LP, in order to generate new simplex multipliers. Although the procedure takes advantage of a previous basis when solving a the system with an added column, the whole procedure is painfully slow, and is only adopted to show that it can be used to solve MD-BPP.

We initially apply the proposed algorithm on a subset of 2D-BPP instances investigated by Martello and Vigo [1998]. This subset contains 28 problems originally created as 2D-KP instances, and are solved as 2D-BPP with fixed orientation. Christofides and Whitlock [1977] generate three instances (CGCUT1-CGCUT3), and Beasley [1985a and 1985b] the other 25 instances (GCUT1-GCUT13 and NGCUT1-GCUT12). Among these instances, on 14 occasions the solution with the oriented algorithm does not equal the continuous lower bound and could benefit from the application of an orthogonal algorithm. In 12 of these instances, the algorithm proposed here obtains an orthogonal solution better than the oriented solution, with a 10.4% (17 bin) reduction in the number of bins required to pack all items.

Table VII.1 presents the data relating 2D-BPP instances analyzed. The first column identifies the instance; the next column shows the number of items to pack in the instance; the third column represents the continuous lower bound; the fourth lists the number of bins required in the oriented version of the instance, as by Martello and Vigo [1998]; the fifth

lists the number of bins required in the orthogonal version, as obtained in this dissertation; and the last column shows the relative reduction in the number of bins used in each instance.

Instance	# Item Types	# Items	Continuous Lower Bound	Martello and Vigo Solution	Orthogonal Optimal Solution	Variation (%)
CGCUT1	7	16	2	2	2	0.0
CGCUT2	10	23	2	2	2	0.0
CGCUT3	20	62	16	23	18	27.8
GCUT1	10	10	3	5	4	25.0
GCUT2	20	20	5	6	5	20.0
GCUT3	30	30	7	8	7	14.3
GCUT4	50	50	12	14	13	7.7
GCUT5	10	10	3	3	3	0.0
GCUT6	20	20	5	7	6	16.7
GCUT7	30	30	9	11	10	10.0
GCUT8	50	50	12	14*	12	16.7
GCUT9	10	10	3	3	3	0.0
GCUT10	20	20	6	7	7	0.0
GCUT11	30	30	7	9	8	12.5
GCUT12	50	50	13	16	15	6.7
GCUT13	32	32	2	2	2	0.0
NGCUT1	5	10	2	3	3	0.0
NGCUT2	7	17	3	4	3	33.3
NGCUT3	10	21	3	3	3	0.0
NGCUT4	5	7	2	2	2	0.0
NGCUT5	7	14	3	3	3	0.0
NGCUT6	10	15	2	3	2	50.0
NGCUT7	5	8	1	1	1	0.0
NGCUT8	7	13	2	2	2	0.0
NGCUT9	10	18	3	3	3	0.0
NGCUT10	5	13	2	3	3	0.0
NGCUT11	7	15	2	2	2	0.0
NGCUT12	10	22	3	3	3	0.0
TOTAL	**487**	**636**	**135**	**164**	**147**	**11.6**

Table VII.1 *Results for 2D-BPP instances solved with our exact algorithm.*
The first column identifies the instance; the next two columns show the number of different item types and number of items to pack in the instance; the fourth column represents the continuous lower bound; the fifth has the number of bins required in the oriented version of the instance, as obtained in Martello and Vigo [1998]; the sixth has the number of bins required in the orthogonal version, as obtained in the present work; and the last column shows the relative increase in the number of bins, when comparing the optimal solution value of the oriented version with the optimal solution value of the orthogonal version.
** Indicates that the result is the best available, but may not be optimal for the oriented version of the instance.*

Observe that the fixed orientation restriction imposed by Martello and Vigo might produce solutions 50% larger than necessary in a packing situation, as in instance NGCUT6. Even in instances requiring the use of several bins, as in CGCUT3, the restriction contributes an increase of 27.8% to the number of bins used.

Run times involved in solving these instances to optimality range from less than a second, as in instance NGCUT7, to several hours, as in instance GCUT8. In this latter instance, more than 200 columns are generated by the algorithm, 136 of those in the initial relaxed LP. Table VII.2 presents the number of columns generated in the procedure, in the initial relaxed LP, and within the branch-and-bound procedure.

Instance	# Item Types	# Items	Continuous Lower Bound	Orthogonal Optimal Solution	# Columns in the Initial Phase	# Columns in the BB Phase
CGCUT1	7	16	2	2	23	0
CGCUT2	10	23	2	2	12	0
CGCUT3	20	62	16	18	59	17
GCUT1	10	10	3	4	20	0
GCUT2	20	20	5	5	58	0
GCUT3	30	30	7	7	97	30
GCUT4	50	50	12	13	121	0
GCUT5	10	10	3	3	13	0
GCUT6	20	20	5	6	64	0
GCUT7	30	30	9	10	80	0
GCUT8	50	50	12	12	136	100
GCUT9	10	10	3	3	13	0
GCUT10	20	20	6	7	44	0
GCUT11	30	30	7	8	79	0
GCUT12	50	50	13	15	128	3
GCUT13	32	32	2	2	34	0
NGCUT1	5	10	2	2	12	5
NGCUT2	7	17	3	3	18	0
NGCUT3	10	21	3	3	25	0
NGCUT4	5	7	2	2	6	0
NGCUT5	7	14	3	3	20	0
NGCUT6	10	15	2	2	29	0
NGCUT7	5	8	1	1	6	0
NGCUT8	7	13	2	2	8	0
NGCUT9	10	18	3	3	31	0
NGCUT10	5	13	2	3	12	0
NGCUT11	7	15	2	2	20	0
NGCUT12	10	22	3	3	34	0

Table VII.2 *Details of column generation in the application of the B&P algorithm to the same instances as in Table VII.1.*
The first column identifies the instance, the next two columns show the number of different item types and number of items to pack in the instance, the fourth column represents the continuous lower bound, the fifth has the number of bins required in the orthogonal version, the sixth column shows the number of columns generated in the initial LP, and the last column presents the number of columns generated within the branch-and-price algorithm.

We also apply the algorithm on 47 3D-BPP instances (Iva-1 to Iva-47) initially investigated by Ivancic et al (as reported by Bischoff and Ratcliff [1995]) and Bischoff and Ratcliff [1995]. Both works consider heuristics for orthogonal 3D-BPP and in these

instances all items can be rotated on the three axes. On 12 instances, one or both heuristics are able to obtain an optimal solution, as verified with the continuous lower bound, leaving 35 instances to be investigated. The exact solution obtained by the new algorithm is better than the previously known solution on 30 instances. In total, the solution with the heuristics requires 79 bins more than the optimal solution. The combined result, selecting the best heuristic solution to each instance, requires 61 more bins or 8.9% of the total.

Table VII.3 presents data for the instances of 3D-BPP analyzed. The first column identifies the instance, the next column shows the number different types of items to pack in the instance, the third column represents the continuous lower bound, the fourth and fifth have the number of bins obtained with the heuristics proposed by Ivancic et al (as reported by Bischoff and Ratcliff [1995]) and Bischoff and Ratcliff, respectively, the sixth column has the number of bins required in the optimal solution to the orthogonal version, as obtained in the present work, and the last column shows the relative increase in the number of bins used in each instance, with respect to the best solution in the previous work.

Observe that the heuristics generate solutions up to 50% larger than the optimal solution.

Run times to solve these instances to optimality range from less than a second, as in instance Iva-1 to more than four days, as in instance Iva-45. In this latter instance, the volume utilization in each of the two bins is larger than 97%, with 47 and 52 items packed. In some cases, the 3D-DFA algorithm requires several hours to generate a new column, with only 17 columns generated, 7 in the initial relaxed LP, and 10 others within the branch-and-bound procedure (Table VII.4).

Instance	# Item Types	# Items	Continuous Lower Bound	Ivancic et al Solution	Bischoff / Ratcliff Solution	Orthogonal Optimal Solution	Change from Previous Best (%)
Iva-1	2	70	25	26	27	25	4.0
Iva-2	2	70	8	11	11	9	22.2
Iva-3	4	180	19	20	21	19	5.3
Iva-4	4	180	26	27	29	26	3.8
Iva-5	4	180	50	65	61	51	19.6
Iva-6	3	103	10	10	10	10	0.0
Iva-7	3	103	16	16	16	16	0.0
Iva-8	3	103	4	5	4	4	0.0
Iva-9	2	110	16	19	19	19	0.0
Iva-10	2	110	37	55	55	55	0.0
Iva-11	2	110	14	18	19	16	12.5
Iva-12	3	95	45	55	55	53	3.8
Iva-13	3	95	20	27	25	25	0.0
Iva-14	3	95	27	28	27	27	0.0
Iva-15	3	95	11	11	11	11	0.0
Iva-16	3	95	21	34	28	26	7.7
Iva-17	3	95	7	8	8	7	14.3
Iva-18	3	47	2	3	3	2	50.0
Iva-19	3	47	3	3	3	3	0.0
Iva-20	3	47	4	5	5	5	0.0
Iva-21	5	95	17	24	24	20	20.0
Iva-22	5	95	8	10	11	8	25.0
Iva-23	5	95	17	21	22	18	16.7
Iva-24	4	72	5	6	6	5	20.0
Iva-25	4	72	4	6	5	5	0.0
Iva-26	4	72	3	3	3	3	0.0
Iva-27	3	95	4	5	5	4	25.0
Iva-28	3	95	9	10	11	9	11.1
Iva-29	4	118	15	18	17	16	6.3
Iva-30	4	118	18	24	24	20	20.0
Iva-31	4	118	11	13	13	12	8.3
Iva-32	3	90	4	5	4	4	0.0
Iva-33	3	90	4	5	5	4	25.0
Iva-34	3	90	7	9	9	8	12.5
Iva-35	2	84	2	3	3	2	50.0
Iva-36	2	84	14	18	19	14	28.6
Iva-37	3	102	22	26	27	23	13.0
Iva-38	3	102	45	50	56	45	11.1
Iva-39	3	102	12	16	16	15	6.7
Iva-40	4	85	7	9	10	8	12.5
Iva-41	4	85	14	16	16	15	6.7
Iva-42	3	90	4	4	5	4	0.0
Iva-43	3	90	3	3	3	3	0.0
Iva-44	3	90	3	4	4	3	33.3
Iva-45	4	99	2	3	3	2	50.0
Iva-46	4	99	2	2	2	2	0.0
Iva-47	4	99	3	4	3	3	0.0
TOTAL	**154**	**4,556**	**624**	**763**	**763**	**684**	**8.9**

Table VII.3 *Results of instances from the literature solved with the exact algorithm for 3D-BPP.*

Combining the best heuristic results, for each instance, a total of 745 bins are needed to pack all items, a number 8.9% larger than the 684 bins required in the optimal solution.

Instance	# Item Types	# Items	Continuous Lower Bound	Orthogonal Optimal Solution	# Columns in the Initial Phase	# Columns in the BB Phase
Iva-1	2	70	25	25	3	0
Iva-2	2	70	8	9	4	3
Iva-3	4	180	19	19	6	5
Iva-4	4	180	26	26	5	2
Iva-5	4	180	50	51	6	0
Iva-6	3	103	10	10	4	0
Iva-7	3	103	16	16	3	0
Iva-8	3	103	4	4	4	0
Iva-9	2	110	16	19	2	1
Iva-10	2	110	37	55	2	0
Iva-11	2	110	14	16	2	1
Iva-12	3	95	45	53	4	0
Iva-13	3	95	20	25	4	2
Iva-14	3	95	27	27	5	0
Iva-15	3	95	11	11	6	0
Iva-16	3	95	21	26	5	0
Iva-17	3	95	7	7	6	2
Iva-18	3	47	2	2	4	0
Iva-19	3	47	3	3	4	0
Iva-20	3	47	4	5	6	8
Iva-21	5	95	17	20	9	1
Iva-22	5	95	8	8	10	7
Iva-23	5	95	17	18	8	11
Iva-24	4	72	5	5	6	10
Iva-25	4	72	4	5	7	3
Iva-26	4	72	3	3	6	0
Iva-27	3	95	4	4	3	15
Iva-28	3	95	9	9	5	6
Iva-29	4	118	15	16	6	7
Iva-30	4	118	18	20	7	5
Iva-31	4	118	11	12	5	14
Iva-32	3	90	4	4	4	3
Iva-33	3	90	4	4	3	12
Iva-34	3	90	7	8	6	10
Iva-35	2	84	2	2	3	1
Iva-36	2	84	14	14	3	0
Iva-37	3	102	22	23	5	0
Iva-38	3	102	45	45	4	0
Iva-39	3	102	12	15	5	3
Iva-40	4	85	7	8	5	9
Iva-41	4	85	14	15	8	3
Iva-42	3	90	4	4	4	0
Iva-43	3	90	3	3	4	0
Iva-44	3	90	3	3	4	1
Iva-45	4	99	2	2	10	7
Iva-46	4	99	2	2	8	0
Iva-47	4	99	3	3	6	0

Table VII.4 *Details of column generation in the application of the B&P algorithm to the same instances as in Table VII.3.*
The first three columns are the same as those in Table VII.3; the fourth column represents the continuous lower bound; the fifth has the number of bins required in the orthogonal version; the sixth column shows the number of columns generated in the initial LP; and the last column presents the number of columns generated within the branch-and-price algorithm.

Recently, Eley [2002] presents results of the application of a new heuristic on, supposedly, the same data set, obtaining complete packing using 716 bins. We are not able to compare our results with those reported by Eley because some differences can be observed between the instances of the data sets used in both works, with Eley reporting solutions to 3D-BPP using a number of bins smaller than the volume ratio bound (continuous lower bound) for the instances investigated here, and available at the OR-Library [Beasley 2002]. As an example, Table VII.5 lists the data for instance Iva-18. The continuous lower bound for this instance is 2, but Eley reports requiring only one bin to pack all items.

# Item Types	3		
Bin Dimensions:	(30, 21, 18)	Volume of Bin:	11,340
Items Dimensions:	(6, 9, 12)	(7, 4, 10)	(3, 5, 11)
# Items Available	20	15	12
Volume of Items	12,960	4,200	1,980
Volume Ratio	1.68		

Table VII.5 *Details of instance Iva-18, initially investigated by Ivancic at al (as reported by Bischoff and Ratcliff [1995]) and available online at the OR-Library [Beasley 2002]. Eley [2002] reports packing all items in this instance in only one bin, while the volume ratio bound (continuous lower bound) is two bins.*

C. A MORE GENERAL VERSION OF 2D-BPP

Up to this point, the problems covered in this chapter belong to the groups $2/V/I/R$ and $3/V/I/R$ in Dyckhoff's typology [1990]. In these two groups, there is only one type of bin available. Another interesting set of instances has bins (or pieces of stock) of different sizes available, identified by $2/V/D/R$ or $3/V/D/R$.

Wang [1983] studies a variation of 2D-BPP where stock rectangles with different sizes are available, and the objective is to minimize the waste in the production a given number of rectangular pieces. She proposes a heuristic based on constructing larger rectangles from smaller ones to solve this problem, and presents the results of applying the algorithm on two instances.

Beasley [1985c] investigates an even more general version of the problem, the *Two-Dimensional Assortment Problem* (*2D-AP*), considering the cost of the different pieces of stock and smaller rectangular items. Following Wang [1983], Beasley solves the instances using a heuristic, with column generation as in the original work of Gilmore and Gomory [1965].

Both authors only consider guillotine cut patterns in their algorithms.

We present two new heuristics, based on replacing the 2D-DFA in the exact algorithm for 2D-BPP. In the former heuristic, we adopt the five-block heuristic for 2D-KP to generate the packing, or cutting, patterns. In the latter, the 2D-LDFA is used.

	Skalbeck and Schultz	**Wang [1983]**	**Beasley [1985c]**	**5B**	**2D-LDFA**
Stock Area	326,736	331,344	327,312	322,992	322,128
Waste (%)	1.90	3.20	2.05	0.74	0.47
Run Time	-	12min 58sec	8min 3sec	12 sec	1min 15sec

Table VII.6 *Results of the application of different heuristics to an instance investigated by Skalbeck and Schultz (as reported by Beasley [1985c]), and also solved by Wang [1983] and Beasley [1985c].*
The "5B" column lists the results of using the five-block heuristic for 2D-KP, and the "2D-LDFA" column lists the results of using the 2D-LDFA heuristic, limited to 1,000 backtracks, instead of the 2D-DFA within the branch-and-price algorithm for 2D-BPP.

The heuristic is initially applied on an instance originally investigated by Skalbeck and Schultz (as reported by Beasley [1985c]), and also solved by Wang [1983] and Beasley [1985c]. Table VII.6 presents the results published in these papers, and two results obtained in this dissertation: the "5B" column lists the results of using the five-block heuristic for 2D-KP, and the "2D-LDFA" column lists the results of using the 2D-LDFA heuristic, limited to 1,000 backtracks, instead of the 2D-DFA, within the branch-and-price algorithm for 2D-BPP.

Beasley [1985c] investigates another set of with 12 instances of 2D-AP. The website of the OR-Library [Beasley 2002] reports that "the optimal solution values are not known at present." We apply the five-block based heuristic on the same instances. The results obtained are listed in Table VII.7.

Instance	Total Area of Items	Total Stock Area	Beasley [1985c]		5B	
			Waste (%)	Run Time (sec)	Waste (%)	Run Time (sec)
1	288,840	304,270	7.69	14.8	5.07	0.6
2	507,080	514,928	4.17	41.1	1.52	1.7
3	812,020	831,262	5.87	91.4	2.31	4.0
4	243,640	249,406	6.63	26.5	2.31	0.4
5	560,040	566,708	4.95	116.7	1.17	3.4
6	840,240	856,391	7.62	313.3	1.89	9.1
7	1,691,060	1,777,783	16.84	12.5	4.88	1.5
8	3,638,720	3,792,048	5.48	58.9	4.03	11.7
9	5,311,440	5,408,779	9.07	116.9	1.80	31.0
10	1,624,940	1,757,942	13.80	24.4	7.57	0.8
11	3,538,860	3,642,437	6.65	204.3	2.84	8.0
12	5,131,900	5,265,505	5.89	182.3	2.54	18.6

Table VII.7 *Results for 12 instances of 2D-AP investigated by Beasley [1985c]. Results credited to Beasley are from the original work, including run times obtained with a CDC 7600 computer. The columns labeled "5B" present the results of applying the five-block based heuristic, on an Athlon 1 GHz personal computer.*

VIII. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

In this chapter, we review the contributions of this dissertation and suggest further research.

A. CONCLUSIONS AND CONTRIBUTIONS

This dissertation covered distinct flavors of Multidimensional Packing Problems, considering instances in two and three dimensions, with different objective functions. In all instances considered here, both items and bins have rectangular shapes.

In **Chapter I**, besides introducing the notation and the typology of MD-PP, and reviewing previous work, we analyze the effect of restricting rotation and certain types of pattern restrictions on the number of bins used on 2D-BPP. It is shown that restricting the orientation of items can lead to solutions using twice as many bins as when 90º rotations are allowed. The restrictions of guillotine cut patterns and 1st-order non-guillotine patterns are also considered, with asymptotic results presented. For the case when items have to be packed together, or connected, up to a 33% increase in the number of bins can be expected.

Chapters III and **IV** are dedicated to Pallet Loading Problems (PLP).

In **Chapter III**, we define the idea of the Minimum Size Instance (MSI) of an equivalence class of PLP, and show that every class has one and only one MSI. This makes the MSI extremely helpful in distinguishing equivalence classes. We also develop bounds on the dimensions of item and pallet in the MSI of a class, given the set of efficient partitions of an instance of that class. We also show that a bound used for almost 15 years [Dowsland 1987] is incorrect.

Applying the newly developed bounds to the MSI, we enumerate all equivalence classes with area ratio bound as large as 100. Previous work in this area considered at most a subset of these classes.

Chapter IV presents a set of new bounds and algorithms developed with the objective of solving all instances enumerated in **Chapter III**.

The first of these new bounds is the Single Homogeneous Perfect Partition (SHPP) bound. This bound, together with simple heuristics, proves optimality of almost 50% of all classes that have no proven optimal solution after applying other simple bounds.

The second bound is the Single Perfect Partition (SPP) bound. It can be useful when the MSI of a class presents only one perfect X-partition or Y-partition.

The third bound is the Relaxed Class (RC) bound, requiring the definition of new relations among distinct equivalence classes of PLP – restricted and relaxed classes. Every feasible packing pattern in a restricted class is shown to be adoptable by a relaxed class.

The fourth bound, the Combined Perfect Partition and Relaxed Class (CPPRC) bound, combines ideas of the SHPP and the RC bounds.

Bounds based on similarity of classes and an improved LP bound are presented, but not completely explored. These two bounds are applied to only a few instances of PLP.

Among the algorithms, the Hollow Block Heuristic is the simplest one proposed in **Chapter IV**. It is related to the diagonal heuristic, described in Nelissen [1993], but has a simpler structure. The G5-heuristic generates 1st-order non-guillotine packing patterns, and finds optimal solutions to all instances of PLP with an area ratio (AR) bound of up to 51 boxes, 99.999% of all instances with an AR bound of up to 100 boxes, and differs at most by one box in the 0.001% remaining instances.

Three other heuristics are proposes based on non-guillotine cut patterns of higher order, with at most eight blocks. These heuristics solve some instances not solved with the G5-heuristic, nor by other heuristics from the literature.

After the application of the proposed bounding procedures and heuristics, 6,952 equivalence classes of PLP with an AR bound of up to 100 boxes remain without a proven optimal solution. An exact algorithm is developed to solve these last remaining instances. The HVZ algorithm is based on the idea of representing a packing pattern with a ternary string, with the characters H, V, and Z. This new exact algorithm, together with the application of the RC bound, is able to identify the optimal solution to the remaining instances, obtaining in only three equivalence classes a solution not generated with the heuristics proposed in this dissertation.

Chapters V and **VI** are dedicated to knapsack problems, in two and three dimensions (2D-KP and 3D-KP).

In **Chapter V** we implement a mixed integer program (MIP) model of 2D-KP and show that it can be used, with a commercial optimization software to solve some instances from the literature.

In **Chapter VI**, we extend the HVZ algorithm for PLP to develop the Diagonal Fill Algorithm (DFA) for 2D-KP and 3D-KP, and show that these new algorithms are able to solve orthogonal versions of instances from the literature only solved before considering fixed orientation or guillotine cut patterns. The two proposed algorithms, 2D-DFA and 3D-DFA, are the first algorithms in the literature to solve orthogonal non-guillotine instances of 2D-KP and 3D-KP.

Multidimensional Bin Packing Problems (MD-BPP) are addressed in **Chapter VII**. New branch-and-price algorithms for 2D-BPP and 3D-BPP, based on the 2D-DFA and 3D-DFA are developed. These are also the first algorithms in the literature to solve the orthogonal non-guillotine instances.

For cases where a good solution soon is better than an optimal solution later, we substitute the exact algorithms for MD-KP in the branch-and-price procedure for MD-BPP. This new heuristic finds better solutions to instances of MD-BPP from the literature.

B. SUGESTIONS FOR FURTHER RESEARCH

In PLP, several open questions remain regarding the representation of packing patterns with HVZ strings. Among these questions, we can list:

- How many distinct HVZ strings are necessary to represent an optimal packing pattern for all instances with up to a given number of items packed?
- Given a PLP instance, is it possible to compute a tight bound on the length of the HVZ string?
- Given an HVZ string, can we determine the classes in which it corresponds to an optimal packing pattern?
- Is there an efficient method of generating the packing pattern from the HVZ string?

The MIP model, analyzed in **Chapter V**, is applied only with a commercial solver, and we do not investigate of the performance of this model with a special-purpose branch and bound algorithm. One such procedure might branch on all indicator variables related to the comparison of two items at a time. Another procedure might fix, at each node, the position of an item, and insert additional constraints on other item positions. Another application of the model, also not explored, is within the 2D- and 3D-DFA. In this case, a relaxed model could compute bounds at some or all nodes of the search tree.

Both the 2D-DFA and 3D-DFA are implemented with the primary purposes of showing that they can effectively solve MD-KP instances of reasonable size and complexity. Several performance improvements can be applied to these algorithms. For example, one deficiency of the algorithms is that the packing solution generated at each step is not reused in the next step, when a new item is included in the packing list. We believe that a significant improvement in processing speed can be achieved by correcting this deficiency.

The algorithms for MD-BPP have been implemented only as prototypes and would require refining for use in the real world.

LIST OF REFERENCES

Adams, D.J., "A Heuristic Procedure to Aggregate Containers onto Pallets and Plan the Loading of Pallets into Trucks," MS Thesis in Operations Research, Naval Postgraduate School, 1996.

Arenales, M. and Morabito, R., "An AND/OR-Graph Approach to the Solution of Two-Dimensional Non-Guillotine Cutting Problems," *European Journal of Operational Research*, Vol. 84, pp. 599-617, 1995.

Baker, B.S., Coffman, E.G., and Rivest, R.L., "Orthogonal Packings in Two Dimensions," *SIAM Journal in Computing*, Vol. 9, pp. 846-855, 1980.

Baker, B.S. and Schwarz, J.S., "Shelf Algorithms for Two-Dimensional Packing problems," *SIAM Journal in Computing*, Vol. 12, pp. 508-525, 1983.

Barnes, F.W., "Packing the Maximum Number of m x n Tiles in a Large p x q Rectangle," *Discrete Mathematics*, Vol. 26, pp. 93-100, 1979.

Barnett, S. and Kynch, G.J., "Exact Solution of a Simple Cutting Problem," *Operations Research*, Vol. 15, pp. 1051-1056, 1967.

Beasley, J.E., "Algorithms for Unconstrained Two-Dimensional Guillotine Cutting," *Journal of the Operational Research Society*, Vol. 36, pp. 297-306, 1985a.

Beasley, J.E., "An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure," *Operations Research*, Vol. 33, pp. 49-64, 1985b.

Beasley, J.E., "An Algorithm for the Two-Dimensional Assortment Problem," *European Journal of Operational Research*, Vol. 19, pp. 253-261, 1985c.

Beasley, J.E., "OR-Library," [http://mscmga.ms.ic.ac.uk/info.html], 2002.

Bertsimas, D. and Tsitsiklis, J.N., *Introduction to Linear Optimization*, Athena Scientific, 1997.

Bhattacharya, S., Roy, R., and Bhattacharya, S., "An Exact Depth-First Algorithm for the Pallet Loading Problem," *European Journal of Operational Research*, Vol. 110, pp. 610-625, 1998.

Biró, M. and Boros, E., "Network Flows and Non-Guillotine Cutting Patterns," *European Journal of Operational Research*, Vol. 16, pp. 215-221, 1984.

Bischoff, E.E. and Dowsland, W.B., "An Application of the Micro to Product Design and Distribution," *Journal of the Operational Research Society*, Vol. 33, pp. 271-281, 1982.

Bischoff, E.E., Janetz, F., and Ratcliff, M.S., "Loading Pallets with Non-identical Items," *European Journal of Operational Research*, Vol. 84, pp. 681-692, 1995.

Bischoff, E.E. and Marriot, M., "A Comparative Evaluation of Heuristics for Container Loading," *European Journal of Operational Research*, Vol. 44, pp. 267-276, 1990.

Bischoff, E.E. and Ratcliff, F., "Issues in the Development of Approaches to Container Loading," *Omega*, Vol. 23, pp. 377-390, 1995.

Bischoff, E.E. and Wäscher, G., "Cutting and Packing (special issue)," *European Journal of Operational Research*, Vol. 44, 1990.

Brooks, R.L., Smith, C.A.B., Stone, A.H., and Tutte, W.T., "The dissection of rectangles into squares," *Duke Mathematical Journal*, Vol. 7, pp. 312-340, 1940.

Brualdi, R.A. and Foregger, T.H., "Packing Boxes with Harmonic Bricks," *Journal of Combinatorial Theory, Series B*, Vol. 17, pp. 81-114, 1974.

CADAP, "CADAP – Cutting and Packing at Dresden University of Technology," [http://www.math.tu-dresden.de/~capad/capad.html], 2002.

Chen, C.S., Lee, S.M., and Shen, Q.S., "An Analytical Model for the Container Loading Problem," *European Journal of Operational Research*, Vol. 80, pp. 68-76, 1995.

Christofides, N. and Hadjiconstantinou, E., "An Exact Algorithm for Orthogonal 2-D Cutting Problems, Using Guillotine Cuts," *European Journal of Operational Research*, Vol. 83, pp. 21-38, 1995.

Christofides, N. and Whitlock, C., "An Algorithm for Two-Dimensional Cutting Problems," *Operations Research*, Vol. 25, pp. 30-45, 1977.

Chung, F.R.K., Garey, M.R., and Johnson, D.S., "On Packing Two-Dimensional Bins," *SIAM Journal on Algebraic and Discrete Methods*, Vol. 3, pp. 66-76, 1982.

Coffman, E.G., Garey, M.R., and Johnson, D.S., "Performance Bounds for Level-Oriented Two-Dimensional Packing Algorithms," *SIAM Journal on Computing*, Vol. 9, pp. 808-826, 1980.

Coffman, E.G. and Shor, P.W., "Average-Case Analysis of Cutting and Packing in Two Dimensions," *European Journal of Operations Research*, Vol. 44, pp. 134-144, 1990.

Coffman, E.G. and Lueker, G.S., *Probabilistic Analysis of Packing and Partitioning Algorithms*, John Wiley & Sons, Inc., New York, 1991.

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R. and Schrijver, A., *Combinatorial Optimization*, John Wiley & Sons, Inc., New York, 1998.

Daza, V.P., Alvarenga, A.G., and Diego, J., "Exact Solutions for Constrained Two-Dimensional Cutting Problems," *European Journal of Operational Research*, Vol. 84, pp. 633-644, 1995.

De Bruijn, N.G., "Filling Boxes with Bricks," *American Mathematics Quarterly*, Vol. 76, pp. 37-40, 1969.

Dowsland, K.A., "The Three-Dimensional Pallet Chart: An Analysis of the Factors Affecting the Set of Feasible Layouts for a Class of Two-Dimensional Packing Problems," *Journal of The Operations Research Society*, Vol. 35, pp. 895-905, 1984.

Dowsland, K.A., "Determining an Upper Bound for a Class of Rectangular Packing Problems," *Computers and Operations Research*, Vol. 12, pp. 201-205, 1985.

Dowsland, K.A., "A Combined Data-Base and Algorithmic Approach to the Pallet-Loading," *Journal of The Operations Research Society*, Vol. 38, pp. 78-84, 1987a.

Dowsland, K.A., "An Exact Algorithm for the Pallet Loading Problem," *European Journal of Operational Research*, Vol. 31, pp. 78-84, 1987b.

Dowsland, K.A., "Efficient Automated Pallet Loading," *European Journal of Operational Research*, Vol. 44, pp. 232-238, 1990.

Dowsland, K.A. and Dowsland, W.B., "Solution Approaches to Irregular Nesting Problems," *European Journal of Operational Research*, Vol. 84, pp. 506-521, 1995.

Dyckhoff, H., "A Typology of Cutting and Packing Problems," *European Journal of Operational Research*, Vol. 44, pp. 145-159, 1990.

Dyckhoff, H. and Wäscher, G., "Cutting and Packing (special issue)," *European Journal of Operational Research*, Vol. 84, 1995.

Eley, M., "Solving Container Loading Problems by Block Arrangement," *European Journal of Operational Research*, Vol. 141, pp. 393-409, 2002.

Erdös, P. and Graham, R.L., "On Packing Squares with Equal Squares," *Journal of Combinatorial Theory, Series A*, Vol. 19, pp. 119-123, 1975.

Faina, L., "A Global Optimization Algorithm for the Three-Dimensional Packing Problem," *European Journal of Operational Research*, Vol. 126, pp. 340-354, 2000.

Farley, A.A., "Selection of Stockplate Characteristics and Cutting Style for Two Dimensional Cutting Stock Situations," *European Journal of Operational Research*, Vol. 44, pp. 239-246, 1990.

Friedman, E., "Packing Unit Squares in Squares: A Survey and New Results," *The Electronic Journal of Combinatorics*, [http://www.combinatorics.org/Surveys/ds7.html], 2000.

Gallian, J.A., *Contemporary Abstract Algebra*, Houghton Mifflin Company, Boston, 1998.

Gambini, I., "A Method for Cutting Squares into Distinct Squares," *Discrete Applied Mathematics*, Vol. 98, pp. 65-80, 1999.

GAMS Development Corporation, "GAMS 2.50 Rev. 118," 1217 Potomac Street, N.W., Washington, DC, 2000.

Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

Gehring, H., Menschner, K., and Meyer, M., "A Computer-Based Heuristic for Packing Pooled Shipment Containers," *European Journal of Operational Research*, Vol. 44, pp. 277-288, 1990.

George, J.A., and Robinson, D.F., "A Heuristic for Packing Boxes into a Container," *Computers and Operations Research*, Vol. 7, pp. 147-156, 1980.

Gilmore, P.C. and Gomory, R.E., "A Linear Programming Approach to the Cutting Stock Problem," *Operations Research*, Vol. 9, pp. 849-859, 1961.

Gilmore, P.C. and Gomory, R.E., "Multistage Cutting Stock Problems of Two and More Dimensions," *Operations Research*, Vol. 13, pp. 94-120, 1965.

Gómez, A. and de la Fuente, D., "Resolution of Strip-Packing Problems with Genetic Algorithms," *Journal of the Operational Research Society*, Vol. 51, pp. 1289-1295, 2000.

Hadjiconstantinou, E. and Christofides, N., "An Exact Algorithm for General, Orthogonal, Two-Dimensional Knapsack Problems," *European Journal of Operational Research*, Vol. 83, pp. 39-56, 1995.

Healy, P. and Moll, R., "A Local Optimization-Based Solution to the Rectangle Layout Problem," *Journal of the Operational Research Society*, Vol. 47, pp. 523-537, 1996.

Herz, J.C., "A Recursive Computational Procedure for Two-Dimensional Stock-Cutting," *IBM Journal of Research and Development*, Vol. 16, pp. 462-469, 1972.

Hopper, E. and Turton, B.C., "An Empirical Investigation of Meta-Heuristic and Heuristic Algorithms for a 2D Packing Problem," *European Journal of Operational Research*, Vol. 128, pp. 34-57, 2001.

IBM – International Business Machines Corporation, "OSL – Optimization Solutions and Library," New Orchard Road, Armonk, NY 10504, 2000.

ILOG S.A., "CPLEX 6.6," 1080 Linda Vista Ave. Mountain View, CA 94043, 2000.

Johnson, E.L., Nemhauser, G.L., and Savelsbergh, W.P., "Progress in Linear Programming-Based Algorithms for Integer Programming: An Exposition," *INFORMS Journal on Computing*, Vol. 12, pp. 2-23, 2000.

Kantorovich, L.V., "Mathematical Methods of Organizing and Planning Production," *Management Science*, Vol. 6, pp. 366-422, 1960.

Kang, M. and Dai, W., "Arbitrary Rectilinear Block Packing Based on Sequence Pair Structure," *Technical Report: USSC-CRL-98-07*, University of California, Santa Cruz, 1998.

Kenyon, C. and Rémila, E., "A Near-Optimal Solution to a Two-Dimensional Cutting Stock Problem," *Mathematics of Operations Research*, Vol. 25, pp. 645-656, 2000.

Kröger, B., "Guillotineable Bin Packing: A Genetic Approach," *European Journal of Operational Research*, Vol. 84, pp. 645-661, 1995.

Leung, J.Y., Tam, T.W., Wong, C.S., Young, G.H., and Chin, F.Y.L., "Packing Squares into a Square," *Journal of Parallel and Distributed Computing*, Vol. 10, pp. 271-275, 1990.

Li, K. and Cheng, K.H., "Complexity of Resource Allocation and Job Scheduling Problems in Partitionable Mesh Connected Systems," *Proceedings of the First Annual IEEE Symposium of Parallel and Distributed Processing*, IEEE Computer Society, pp. 358-365, 1989.

Li, K. and Cheng, K.H., "On Three-Dimensional Packing," *SIAM Journal on Computing*, Vol. 19, pp. 847-867, 1990.

Li, K. and Cheng, K.H., "Heuristic Algorithms for On-Line Packing in Three Dimensions," *Journal of Algorithms*, Vol. 13, pp. 589-605, 1992.

Lin, E.Y., "A Bibliographical Survey on Some Well-Known Non-Standard Knapsack Problems," *Information Systems and Operational Research*, Vol. 36, pp. 274-317, 1998.

Lins, L., Lins, S., and Morabito, R. "An N-tet Graph Approach for Non-Guillotine Packings of N-Dimensional Boxes into an N-Container," *European Journal of Operational Research*, Vol. 141, pp. 421-439, 2002.

Lodi, A., Martello, S., and Monaci, M., "Two-Dimensional Packing Problems: A Survey," *European Journal of Operational Research*, Vol. 141, pp. 241-252, 2002a.

Lodi, A., Martello, S., and Vigo, D., "Heuristic Algorithms for the Three-Dimensional Bin Packing Problem," *European Journal of Operational Research*, Vol. 141, pp. 410-420, 2002b.

Martello, S., Pisinger, D., and Vigo, D., "The Three-Dimensional Bin Packing Problem," *Operations Research*, Vol. 48, pp. 256-267, 2000.

Martello, S. and Vigo, D., "Exact Solution of the Two-Dimensional Finite Bin Packing Problem," *Management Science*, Vol. 44, pp. 388-399, 1998.

Meir, A. and Moser, L., "On Packing of Squares and Cubes," *Journal of Combinatorial Theory*, Vol. 5, pp. 126-134, 1968.

Meschkowski, H., *Unsolved and Unsolvable Problems in Geometry*, Frederick Ungar Publishing Co., New York, 1966.

Microsoft Corporation, "Microsoft Excel 2000," 1 Microsoft Way, Redmond, WA, 1999.

Miyazawa, F.K. and Wakabayashi, Y., "An Algorithm for the Three-Dimensional Packing Problem," *Algorithmica*, Vol. 18, pp. 122-144, 1997.

Miyazawa, F.K. and Wakabayashi, Y., "Approximation algorithms for the Orthogonal Z-Oriented 3-D Packing Problem," *SIAM Journal on Computing*, Vol. 29, pp. 1008-1029, 2000.

Moon, J.W. and Moser, L., "Some Packing and Covering Theorems," *Colloquium Mathematicum*, Vol. 17, pp. 103-110, 1967.

Morabito, R. and Morales, S., "A Simple and Effective Recursive Procedure for the Manufacturer's Pallet Loading Problem," *Journal of Operational Research Society*, Vol. 49, pp. 819-828, 1998.

Murata, H., Fujiyoshi, K., Nakatake, S., and Kajitani, Y., "Rectangular-Packing-Based Module Placement," *Proceedings of the IEEE international Conference on Computer-Aided Design*, pp. 472-479, 1995.

Nelissen, J., "New Approaches to the Pallet Loading Problem," Working paper, RWTH Aachen [ftp://ftp.informatik.rwth-aachen.de/pub/reports/pallet.ps.Z], 1993.

Nelissen, J., "Solving the Pallet Loading Problem More Efficiently," Working paper, Graduiertenkolleg Informatik und Technik, Aachen, 1994.

Nelissen, J., "How to Use Structural Constraints to Compute an Upper Bound for the Pallet Loading Problem," *European Journal of Operational Research*, Vol. 84, pp. 662-680, 1995.

Nemhauser, G.L. and Wolsey, L.A., *Integer and Combinatorial Optimization*, John Wiley & Sons, Inc, 1988.

Oliveira, J.F. and Ferreira, J.S., "An Improved Version of Wang's Algorithm for Two-Dimensional Cutting Problems," *European Journal of Operational Research*, Vol. 44, pp. 256-266, 1990.

Pisinger, D., "Heuristics for the Container Loading Problem," *European Journal of Operational Research*, Vol. 141, pp. 382-392, 2002.

Scheithauer, G., personal communication with the author, 2000.

Scheithauer, G. and Sommerweiss, U., "4-Block Heuristic for the Rectangle Packing Problem," *European Journal of Operational Research*, Vol. 108, pp. 509-526, 1998.

Scheithauer, G. and Terno, J., "The G4-Heuristic for the Pallet Loading Problem," *Journal of the Operational Research Society*, Vol. 47, pp. 511-522, 1996.

SICUP, online library, [http://www.apdio.pt/sicup/Sicuphomepage/library.htm], 2002.

Smith, A. and De Cani, P., "An Algorithm to Optimize the Layout of Boxes in Pallets," *Journal of the Operational Research Society*, Vol. 31, pp. 573-578, 1980.

Steudel, H.J., "Generating Pallet Loading Patterns: a Special Case of the Two-Dimensional Cutting Stock Problem," *Management Science*, Vol. 25, pp. 997-1004, 1979.

Sweeney, P.E. and Paternoster, E.R., "Cutting and Packing Problems: A Categorized, Application-Oriented Research Bibliography," *Journal of the Operational Research Society*, Vol. 43, pp. 691-706, 1992.

Terno, J., Scheithauer, G., Sommerweiss, U., and Riehme, J., "An Efficient Approach for the Multi-Pallet Loading problem," *European Journal of Operational Research*, Vol. 123, pp. 372-381, 2000.

Tarnowski, A.G., Terno, J., and Scheithauer, G., "A Polynomial Time Algorithm for the Guillotine Pallet Loading Problem," *Information Systems and Operational Research*, Vol. 32, pp. 275-287, 1994.

Wang, P.Y., "Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems," *Operations Research*, Vol. 31, pp. 573-586, 1983.

Wang, P.Y. and Wäscher, G., "Cutting and Packing (special issue)," *European Journal of Operational Research*, Vol. 141, 2002.

Xu, J., Guo, P.N., Cheng, C.K., "Cluster Refinement for Block Placement," *Proceedings of the 34th Design Automation Conference*, pp. 762-765, 1997.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Associate Professor Robert F. Dell
   Naval Postgraduate School
   Monterey, CA

4. Distinguished Professor Gerald G. Brown
   Naval Postgraduate School
   Monterey, CA

5. Associate Professor Craig W. Rasmussen
   Naval Postgraduate School
   Monterey, CA

6. Professor Alan R. Washburn
   Naval Postgraduate School
   Monterey, CA

7. Professor R. Kevin Wood
   Naval Postgraduate School
   Monterey, CA

8. CDR Gustavo Martins
   Rua Gaviao Peixoto, 288 / 703
   Niteroi, RJ, BRAZIL

9. Captain Scott Frickstein, USAF
   Assistant Professor, Mathematical Sciences
   2354 Fairchild Drive, Suite 6D2A
   USAF Academy, Colorado

10. Estado Maior da Armada
    Attn: Brazilian Naval Commission
    5130 MacArthur Blvd NW
    Washington, DC

11. Diretoria de Abastecimento da Marinha
Attn: Brazilian Naval Commission
5130 MacArthur Blvd NW
Washington, DC

12. Centro de Análises de Sistemas Navais
Attn: Brazilian Naval Commission
5130 MacArthur Blvd NW
Washington, DC